DOI: https://doi.org/10.26565/2304-6201-2022-54-02

УДК 004.891 : 004.9:33

Дослідження застосування алгоритмів штучного інтелекту в системах виявлення/запобігання вторгнень

Т. С. Дейнега, І. І. Сватовський

Дейнега Тарас Сергійович
студент
Харківський Національний Університет ім. В. Н. Каразіна
майдан Свободи 6, 61022, Харків
e-mail: xa11867679@student.karazin.ua ;
*https://orcid.org/*0000-0002-7659-9721

Сватовський Ігор Іванович
к.т.н., доцент
Харківський Національний Університет ім. В.Н. Каразіна
майдан Свободи 6, 61022, Харків
e-mail: i.svatowsky@karazin.ua;
https://orcid.org/0000-0002-1836-5599

Проведено аналіз необхідності та доцільності використання алгоритмів та технологій штучного інтелекту на основі нейронних мереж та нечіткої логіки у системах виявлення та запобігання вторгнень у мережу. Сучасні атаки на мережу відрізняються здатністю змінювати свої характеристики та способи дії майже у реальному часі. Застарілі експертні системи захисту мережі, які засновані на понятті «правило-дія», вже не можуть впоратися з даними видами атак, тому що їм потрібен певний час на обробку інформації про нову атаку та занесення її до своєї бази даних. У роботі пропонується модель системи виявлення/запобігання вторгнень на основі використання нейронної мережі, що навчається на тестовій вибірці, яка створюється за алгоритмами нечіткої логіки. Алгоритм навчання нейронної мережі заснований на методі навчання з вчителем та методі зворотного поширення помилки. Таким чином, повна процедура навчання нейронної мережі вимагає від користувача мати лише дамп перехопленого мережевого трафіку для його подальшої обробки згідно алгоритму створення тестової вибірки. Результати оцінки і практичного тестування запропонованої моделі показують, що подібна схема захисту мережі від атак може працювати досить надійно і використовуватись в якості системи виявлення/запобігання вторгнень для локальних та глобальних мереж.

Ключові слова: *штучний інтелект, системи виявлення/запобігання вторгнень, мережеві пакети, нейронна мережа, алгоритм навчання.*

Research of using the artificial intelligence algorithms in intrusion detection/prevention systems

T. S. Deineha, I. I. Svatovskiy

Taras Deineha
Student
V. N. Karazin Kharkiv National University
6 Svobody Sq., Kharkiv, 61022, Ukraine
e-mail: xa11867679@student.karazin.ua ;
*https://orcid.org/*0000-0002-7659-9721

Igor Svatovskiy
Ph.D., Associate Professor
V. N. Karazin Kharkiv National University
6 Svobody Sq., Kharkiv, 61022, Ukraine
e-mail: i.svatowsky@karazin.ua;
https://orcid.org/0000-0002-1836-5599

The analysis of the necessity and expediency of using artificial intelligence algorithms and technologies based on neural networks and fuzzy logic in systems for detecting and preventing network intrusions has been carried out. Modern network attacks are distinguished by the ability to change their characteristics and modes of action almost in real time. Outdated expert network protection systems based on the concept of "rule-action" can no longer cope with these types of attacks, because they need a certain time to process information about a new attack and store it into their database. The paper proposes a model of an intrusion detection/prevention system based on the use of a neural network trained on a test sample created by using fuzzy logic algorithms.

The learning algorithm of the neural network is based on the method of learning with a teacher and the method of backpropagation of the error. Thus, for the complete neural network training procedure the user only needs to have a dump of the intercepted network traffic for further processing according to the test sample creation algorithm. The results of evaluation and practical testing of the proposed model show that such a network protection scheme can work quite reliably and can be used as an intrusion detection/prevention system for local and global networks.

Keywords: *artificial intelligence, intrusion detection/prevention systems, network packets, neural network, learning algorithm.*

1 Вступ

Аналіз стану безпеки у сфері інформаційно-комунікаційних технологій, який постійно проводиться відомими глобальними компаніями, свідчить про те, що світовий ландшафт загроз постійно змінюється. Так, у щорічному звіті Агентства ЄС з кібербезпеки (ENISA) Threat Landscape 2022 однією з головних загроз вважаються програми-вимагачі, а фішинг зараз визначено як найпоширеніший початковий вектор таких атак. Іншими найбільш суттєвими загрозами є атаки на доступність, які також називаються розподіленими атаками на відмову в обслуговуванні (DDoS). Окрім цього з'являється ширший спектр векторів атак, таких як експлойти нульового дня, дезінформація та глибокі фейки за допомогою штучного інтелекту (ШІ). У результаті з'являються ще більш зловмисні та поширені атаки, які мають і більший руйнівний вплив - розвинуті постійні загрози (APT).

Традиційна мережева безпека ґрунтується на застосуванні статистичного аналізу стану безпеки у вузлах на базі таких пристроїв, як брандмауери, системи виявлення вторгнень (IDS) і системи запобігання вторгненням (IPS), системи антивірусного забезпечення. Однак це є втіленням методології пасивного захисту, яка стає все менше придатною для захисту систем від нових загроз мережевій безпеці. Тому в сучасному світі з'явились нові підходи до побудови систем захисту, які базуються на таких галузях, що швидко розвиваються, як технології штучного інтелекту. До них належать інтелектуальний аналіз даних, машинне навчання, нейронні мережі, нечітка логіка, генетичні алгоритми, опорні векторні машини, дерево рішень та ще інші [1,2].

Алгоритми ШІ можуть запобігти деяким новим загрозам завдяки своїй сутності аналізу загроз безпеці, зловмисного програмного забезпечення та методів протидії їм, та зазвичай розвиваються на основі консолідації та категорування знань про попередні атаки, реалізації загроз і наслідків впливу зловмисного програмного забезпечення, що стались. Такі інструменти ШІ як експертні системи, інтелектуальні агенти, штучні імунні системи, машинне навчання, розпізнавання образів, нечітка логіка, евристика тощо все частіше застосовуються для виявлення та протидії сучасним кібератакам [3].

Використання ШІ може допомогти розширити можливості та ефективність застосування і традиційних методів і засобів захисту, зокрема від складних мережевих атак. Однією з найдинамічніших технологій ШІ вважається технологія штучних нейронних мереж. Вони дозволяють вирішувати широкий спектр практичних технічних завдань з детектування різноманітних проявів мережевих атак [4].

2 Дослідження технологій нейронних мереж

Нейронна мережа імітує людський мозок, створюючи систему обробки інформації, яка складається з великої кількості взаємопов'язаних вузлів (нейронів), що взаємодіють один з одним для вирішення конкретного завдання [1]. Рішення про вихід кожного вузла мережі зважується і обробляється для подачі на вхід для всіх інших вузлів в наступному шарі. Процес навчання, який реалізується нейронною мережею, дозволяє виявляти складні і нелінійні відносини між даними [8]. На рисунку 1 показана типова архітектура нейронної мережі.

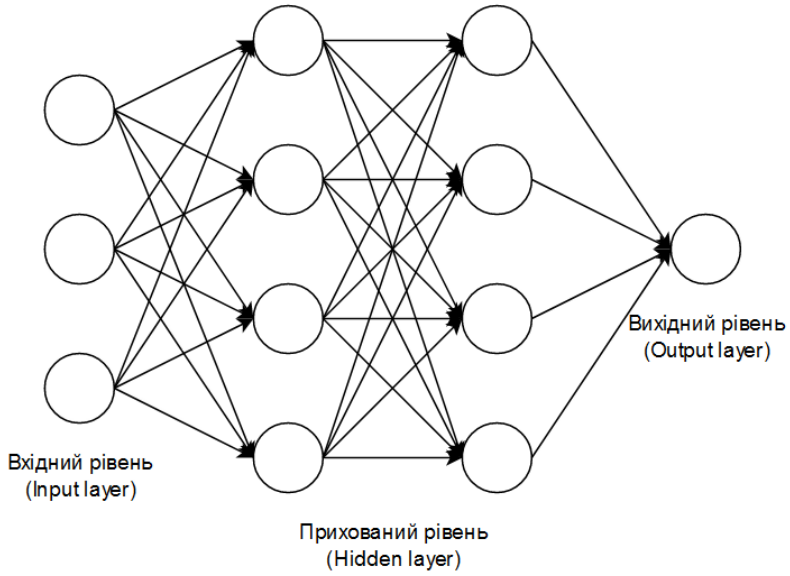

Рис. 1 Архітектура нейронної мережі

Нейронні мережі можуть аналізувати та виявляти відносини і закономірності в даних, а також навчатись на зібраних даних і отриманих знаннях. Для отримання поставлених завдань нейронна мережа повинна бути правильно спроектована. Головна перевага нейронної мережі - здатність робити висновки із зібраних даних без будь-яких попередніх знань про закономірності цих даних, а також її здатність узагальнювати вивчені дані, які дозволяють нейронній мережі виявляти і класифікувати невідомі атаки і різні типи відомих атак. Крім того, як тільки атака розпізнана, нейронна мережа не може бути схильна до цієї атаки у майбутньому. Висока швидкість обробки даних також є однією з переваг нейронної мережі. Щоб успішно впровадити цей метод для системи виявлення та запобігання вторгнень, дані, що представляють нормальну і ненормальну поведінку мережі, повинні бути введені в неї, щоб вона могла автоматично налаштовувати мережеві коефіцієнти на етапі навчання [2, 3].

За останні кілька років було проведено багато досліджень застосування нейронних мереж для виявлення вторгнень [4]. Нейронна мережа зворотного поширення використовується для виявлення вторгнення в систему через її здатність швидкого розпізнавання шаблонних пакетів, отриманих з мережі. Особливості багатьох атак були витягнуті й проаналізовані з стандартних і незвичайних пакетів. Проаналізовані результати цих пакетів були використані для навчання нейронної мережі за зразком пакетів обох типів, використовуючи стандартний алгоритм зворотного поширення помилки.

Щоб подолати низький рівень виявлення, високу кількість помилкових спрацьовувань і інших дефектів, був розроблений новий алгоритм виявлення вторгнень на основі нечіткої нейронної мережі. Цей алгоритм класифікує об'єкти і розпізнає їх нормальну і ненормальну поведінку. В ході великої кількості експериментів продемонстровано, що запропонована модель ефективна і має краще узагальнення. Крім того, швидкість правильного виявлення вторгнень збільшується, а ймовірність помилкового виявлення знижується [1].

3 Практична реалізація технології нейронної мережі на мові програмування Java

Головною метою цієї роботи була реалізація алгоритму нейронної мережі для проведення класифікації мережевого трафіку у дослідній локальній мережі як загрозливого, безпечного чи трафіку, що потребує додаткової перевірки людиною (середній рівень загрози між безпечним та загрозливим). Для дослідження було запропоновано використання алгоритму підготовки тестової вибірки даних для нейронної мережі, алгоритму навчання мережі та застосування навченої нейронної мережі на реальних даних з розрахунком відсотку правильних відповідей та кількості помилок у залежності від ітерації на тестовій вибірці даних [6].

Алгоритм підготовки тестової вибірки для нейронної мережі, заснований на алгоритмах нечіткої логіки (Fuzzy Logic), приймає на вхід текстовий файл з пакетною інформацію, яка була перехоплена у локальній мережі за допомогою сніферу пакетів WireShark [5] та видає на виході тестову вибірку для нейронної мережі у вигляді пар вхідних параметрів та правильної відповіді. Перше значення пари – це масив вхідних даних про мережевий пакет, його розмір дорівнює трьом. До цих даних відноситься: розмір пакету в кілобайтах, наявність у пакеті скриптів та використання мережевих утиліт у пакеті. Друге значення пари – правильна відповідь щодо загрози пакету.

Реалізація алгоритму створення тестової вибірки наведена далі (рис. 2 та 3):

```
private void addPairsToTDataset(final ArrayList<String> packets,
                                final List<Pair<List<Double>, Integer>> dataset) {
    final Pattern pattern = Pattern.compile(PACKET_LENGTH_REGEX);

    for (Iterator<String> iterator = packets.iterator(); iterator.hasNext(); ) {
        final String packet = iterator.next();
        final Matcher matcher = pattern.matcher(packet);
        final List<Double> inputParameters = new ArrayList<>(Arrays.asList(0.0, 0.0, 0.0));
        int outputValue = 0;

        if (packet.contains(PASSWORD) || packet.contains(LOGIN) || packet.contains(CREDENTIALS)) {
            logger.logging( str: FORBIDDEN_WORDS_DETECTED + packet);
            inputParameters.set(0, 1.0);
            outputValue = 1;
        } else if (packet.contains(TCP_PROTOCOL) && packet.contains(SOURCE_AND_DESTINATION_IPS)
                && packet.contains(PORT23) && !packet.contains(TELNET)
                && !packet.contains(DUP) && !packet.contains(OUT_OFF)) {
            logger.logging( str: TELNET_REQUEST_DETECTED + packet);
            inputParameters.set(1, 1.0);
            outputValue = 2;
        } else if (packet.contains(TCP_PROTOCOL) && packet.contains(SOURCE_AND_DESTINATION_IPS) && !packet.contains(SSH)
                && packet.contains(PORT22) && !packet.contains(DUP) && !packet.contains(OUT_OFF)) {
            logger.logging( str: SSH_REQUEST_DETECTED + packet);
            inputParameters.set(1, 1.0);
            outputValue = 2;
```

Рис. 2 Алгоритм створення тестової вибірки даних

```
        } else if (packet.contains(TCP_PROTOCOL) && packet.contains(SOURCE_AND_DESTINATION_IPS) && !packet.contains(SSH)
                && packet.contains(PORT22) && !packet.contains(DUP) && !packet.contains(OUT_OFF)) {
            logger.logging( str: SSH_REQUEST_DETECTED + packet);
            inputParameters.set(1, 1.0);
            outputValue = 2;
        } else if (packet.contains(TCP_PROTOCOL) && packet.contains(SOURCE_AND_DESTINATION_IPS) && !packet.contains(FTP)
                && packet.contains(PORT21) && !packet.contains(DUP) && !packet.contains(OUT_OFF)) {
            logger.logging( str: FTP_REQUEST_DETECTED + System.lineSeparator() + packet);
            inputParameters.set(1, 1.0);
            outputValue = 2;
        }
        while (matcher.find()) {
            final double data = Integer.parseInt(matcher.group());
            if (data > 500) {
                logger.logging( str: LARGE_LENGTH_DETECTED + packet);
                outputValue = 2;
            }
            if (outputValue == 0) {
                dataset.add(Pair.of(List.of(0.0, 0.0, data * BYTE_TO_KILOBYTE), outputValue));
            } else {
                dataset.add(Pair.of(List.of(inputParameters.get(0), inputParameters.get(1),
                        data * BYTE_TO_KILOBYTE), outputValue));
            }
        }
        iterator.remove();
    }
}
```

Рис. 3 Алгоритм створення тестової вибірки даних

Після того, як навчальна вибірка буде зібрана, запускається в роботу алгоритм навчання нейронної мережі. Даний алгоритм заснований на трьох частинах: пряме розповсюдження помилки (forward propagation), зворотне поширення помилки (backward propagation) та оновлення вагових коефіцієнтів зв'язку між нейронами [8]. Код програми для прямого розповсюдження помилки можна побачити на рисунках 4 – 6. Код програми для зворотного поширення помилки можна побачити на рисунках 7 - 9. Код програми для оновлення вагових коефіцієнтів можна побачити на рисунку 10. Шляхом проведення практичних експериментів було знайдено наступні гіпер-параметри для нейронної мережі (Табл. 1):

Табл.1 Гіпер-параметри дослідної нейронної мережі

Назва параметру	Значення
Кількість шарів нейронної мережі	3
Кількість нейронів на другому шарі	10
Кількість ітерацій по навчальній вибірці	1000
Швидкість навчання	0.001

```
final DoubleMatrix t1 = x.mmul(W1).add(b1);
final DoubleMatrix h1 = RELU(t1);
final DoubleMatrix t2 = h1.mmul(W2).add(b2);
final DoubleMatrix result = softMax(t2);
```

Рис.4 Частина коду forward propagation

```
2 usages
private static DoubleMatrix RELU(final DoubleMatrix t) {
    return t.maxi( v: 0, t);
}

2 usages
private static DoubleMatrix softMax(final DoubleMatrix t) {
    final DoubleMatrix out = MatrixFunctions.exp(t);
    return out.divi(out.sum());
}
```

Рис. 5 Функція активації RELU та softMax

```
1 usage
private static double sparseCrossEntropy(final DoubleMatrix result, final int y) {
    return -Math.log(result.get( rowIndex: 0, y));
}
```

Рис. 6 Розрахунок помилки за допомогою розрідженої крос-ентропії

```
private static DoubleMatrix toFull(final int y, final List<String> classification) {
    final DoubleMatrix yFull = DoubleMatrix.zeros( rows: 1, classification.size());
    yFull.put( rowIndex: 0, y,  value: 1);
    return yFull;
}
```

Рис. 7 Функція toFull

```
final DoubleMatrix yFull = toFull(y, classification);
final DoubleMatrix dE_dt2 = result.sub(yFull);
final DoubleMatrix dE_dW2 = h1.transpose().mmul(dE_dt2);
final DoubleMatrix dE_db2 = dE_dt2;
final DoubleMatrix dE_dh1 = dE_dt2.mmul(W2.transpose());
final DoubleMatrix dE_dt1 = dE_dh1.mmul(relu_deriv(t1));
final DoubleMatrix dE_dW1 = x.transpose().mmul(dE_dt1);
final DoubleMatrix dE_db1 = dE_dt1;
```

Рис. 8 Частина коду backward propagation

```
1 usage
private static float relu_deriv(final DoubleMatrix vector) {
    return vector.sum() >= 0 ? 1 : 0;
}
```

Рис. 9 Похідна функція активації relu_deriv

```
W1 = W1.sub(dE_dW1.mul(LEARNING_SPEED));
b1 = b1.sub(dE_db1.mul(LEARNING_SPEED));
W2 = W2.sub(dE_dW2.mul(LEARNING_SPEED));
b2 = b2.sub(dE_db2.mul(LEARNING_SPEED));
```

Рис. 10 Оновлення вагових коефіцієнтів нейронної мережі

Повний фінальний код для навчання нейронної мережі заснований на методі навчання з учителем та на основі алгоритму зворотного поширення помилки представлено на рисунку 11:

```
for (int j = 0; j < 100; j++) {
    for (Pair<List<Double>, Integer> listIntegerPair : dataset) {
        final List<Double> inputVector = listIntegerPair.getLeft();
        double[] inputParameters = new double[]{inputVector.get(0), inputVector.get(1), inputVector.get(2)};
        final DoubleMatrix x = new DoubleMatrix( newRows: 1, newColumns: 3, inputParameters);
        final int y = listIntegerPair.getRight();
        System.out.println(x + System.lineSeparator() + y);

        final DoubleMatrix t1 = x.mmul(W1).add(b1);
        final DoubleMatrix h1 = RELU(t1);
        final DoubleMatrix t2 = h1.mmul(W2).add(b2);
        final DoubleMatrix result = softMax(t2);

        final double error = sparseCrossEntropy(result, y);

        final DoubleMatrix yFull = toFull(y, classification);
        final DoubleMatrix dE_dt2 = result.sub(yFull);
        final DoubleMatrix dE_dW2 = h1.transpose().mmul(dE_dt2);
        final DoubleMatrix dE_db2 = dE_dt2;
        final DoubleMatrix dE_dh1 = dE_dt2.mmul(W2.transpose());
        final DoubleMatrix dE_dt1 = dE_dh1.mmul(relu_deriv(t1));
        final DoubleMatrix dE_dW1 = x.transpose().mmul(dE_dt1);
        final DoubleMatrix dE_db1 = dE_dt1;

        W1 = W1.sub(dE_dW1.mul(LEARNING_SPEED));
        b1 = b1.sub(dE_db1.mul(LEARNING_SPEED));
        W2 = W2.sub(dE_dW2.mul(LEARNING_SPEED));
        b2 = b2.sub(dE_db2.mul(LEARNING_SPEED));
        errors.add(error);
    }
}
```

Рис. 11 Фінальний вигляд алгоритму навчання нейронної мережі

Отримана нейронна мережа має 3 шари (вхідний, другий шар з десяти нейронів та вихідний). Усі нейрони одного рівня пов'язані з нейронами наступного рівня. Схематично дану нейронну мережу можна зобразити наступним чином (рис. 12):

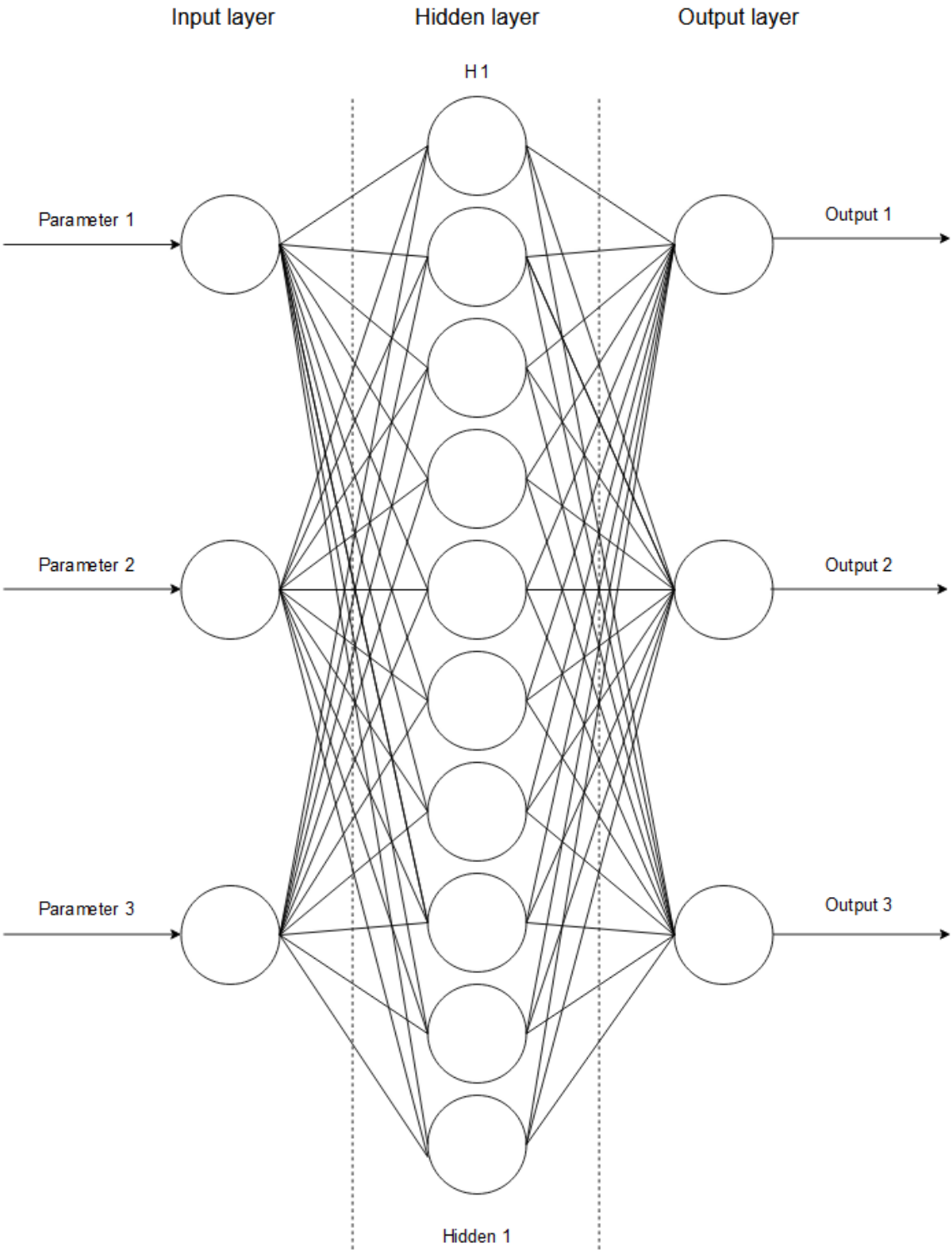

Рис. 12 Схема дослідної нейронної мережі

Після завершення етапу навчання у консолі середовища розробки IntelliJ IDEA Community Edition 2021.3 можна отримати ітерацію по кожній парі навчальної вибірки, а також відсоток правильних відповідей нейронної мережі на останньому циклі проходу по навчальній вибірці (рис. 13), який розраховується таким чином:

```
[0.000000, 0.000000, 0.041016]
0
[0.000000, 0.000000, 0.041016]
0
[0.000000, 0.000000, 0.044922]
0
[0.000000, 0.000000, 0.044922]
0
[0.000000, 0.000000, 0.041016]
0
[0.000000, 0.000000, 0.041016]
0
[0.000000, 0.000000, 0.041016]
0
[0.000000, 0.000000, 0.041016]
0
[0.000000, 0.000000, 0.072266]
0
Accuracy: 0.9844357976653697
```

Рис. 13 Відсоток правильних відповідей нейронної мережі на навчальній вибірці

Як можна бачити з рисунку 13 - відсоток правильних відповідей складає 98.44%, що характеризує мережу як достатньо ефективну. Також після виконання процедур згідно алгоритму навчання можна отримати залежності помилки відповіді від ітерації:

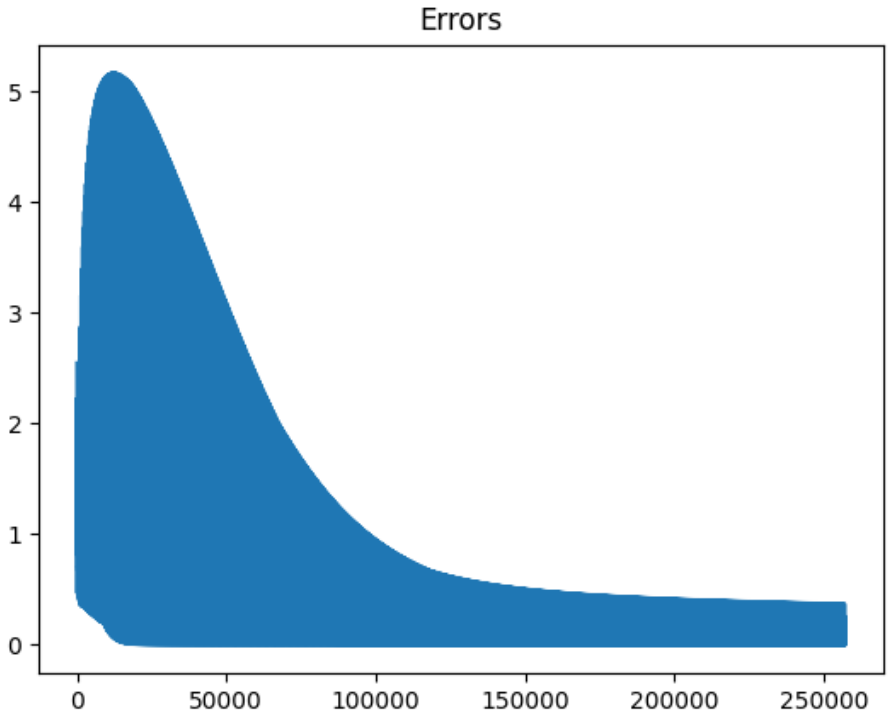

Рис. 14 Графік залежності помилки відповіді від ітерації

Як можна бачити з графіку на рисунку 14 - кількість помилок зменшується зі збільшенням числа пройдених ітерацій. Стрімке зниження кількості помилок спостерігається після 20000 ітерацій алгоритму.

Після того, як нейронна мережа була успішно навчена, у консоль виводяться матриці вагових коефіцієнтів W1 та W2. Їх значення наступні:

- W1 = {0.155353, 0.848620, -0.058468, 0.649779, 0.413327, 0.594318, 0.085725, 0.792108, -0.043930, 0.496328; -0.154272, 0.449024, 0.231025, -0.216459, 1.486742, 0.779725, 1.256136, 0.576876, 0.852188, 0.200115; 0.577778, 0.420876, 0.438282, 0.021567, 1.871525, 0.422121, 1.624109, 0.890067, 0.570674, 0.683717}
- W2 = {0.653276, 0.313120, 0.187101; 0.773230, 0.899240, 0.504340; 0.777735, 0.305611, 0.144749; 1.484853, 0.283755, 0.228076; -0.820227, 0.304082, 1.926313; 0.207484, 0.740289, 0.990918; -0.120909, 0.094986, 1.811141; 0.019589, 0.410993, 1.024896; 0.193575, -0.076539, 1.092565; 0.977203, 0.157549, 0.870955}

Коли система навчена та має правильні вагові коефіцієнти можна протестувати її на реальних даних (мережевих пакетах). У якості реальних пакетів було зібрано інформацію з 20 пакетів мережевого трафіку, у яких містяться і загрозливі пакети даних, і безпечні, і ті, які потребують додаткової перевірки адміністратором мережі (Табл. 2).

Табл.2 Набір параметрів реальних мережевих пакетів

Номер пакету	Перший вхідний параметр	Другий вхідний параметр	Третій вхідний параметр	Правильна відповідь
1	1.0	0.0	0.0546875	1
2	1.0	0.0	0.0566875	1
3	1.0	0.0	0.0786873	1
4	1.0	0.0	0.7046875	1
5	0.0	0.0	0.0125682	1
6	0.0	0.0	0.0855589	0
7	0.0	0.0	0.9858965	2
8	0.0	1.0	0.0458796	2
9	0.0	1.0	0.0745896	2
10	0.0	1.0	0.0546875	2
11	0.0	0.0	0.0658754	0
12	0.0	0.0	0.8546875	2
13	0.0	0.0	0.9658412	2
14	0.0	0.0	0.7485698	2
15	0.0	0.0	0.0658745	0
16	0.0	1.0	0.0452589	2
17	1.0	0.0	0.0745879	1
18	0.0	1.0	0.6857895	2
19	0.0	1.0	0.0546875	2
110	1.0	1.0	0.9856325	2

```
Predication: Need clarify - Correct
Predication: Need clarify - Correct
Predication: Need clarify - Correct
Predication: Not secure - Incorrect
Predication: Secure - Incorrect
Predication: Secure - Correct
Predication: Secure - Correct
Predication: Not secure - Correct
Predication: Not secure - Correct
Predication: Not secure - Correct
Predication: Secure - Correct
Predication: Secure - Correct
Predication: Not secure - Correct
Predication: Not secure - Correct
Predication: Not secure - Correct
Predication: Not secure - Correct
Predication: Need clarify - Correct
Predication: Not secure - Correct
Predication: Not secure - Correct
Predication: Not secure - Correct
Real accuracy: 0.9
```

Рис. 15 Відсоток правильних відповідей нейронної мережі на реальній вибірці

Нейронною мережею було надано правильні відповіді для близько 90% мережевих пакетів (рис. 15). Даний показник трохи відрізняється від того, що був отриманий на тестовій вибірці через те, що у якості реальних даних були обрані також пакети даних, що містять у собі більше одного параметра, який вказує на потенційну загрозу у пакеті. Це суттєво відрізняється від тестової вибірки, яка містить у собі переважно пакети з одним таким параметром.

4 Висновки

Таким чином, у роботі розглянуто та практично реалізовано одну з технологій штучного інтелекту на основі використання нейронної мережі для класифікації мережевого трафіку. Можна з упевненістю сказати, що запропонована модель нейронної мережі продемонструвала достатню ефективність у виявленні та запобіганні мережевих атак. Дана модель має переваги перед традиційними експертними системами та перед системами, заснованими на інших технологіях штучного інтелекту, у швидкодії та відсотку правильних висновків. На реальних даних нейронна мережа показала оціночний результат на рівні близько 90% правильних відповідей та 98% - на тестових даних. Розглянута модель системи виявлення/запобігання вторгнень може буде вдосконалена шляхом додавання до навчальної вибірки більш складних випадків, що наближають склад аналізованого трафіку до умов передачі в реальних мережах. Для адаптації процесу навчання мережі до таких умов доцільно застосувати тестові вибірки, складені з пакетів, які мають декілька атрибутів, що можуть свідчити про наявність в них тих чи інших актуальних загроз.

ЛІТЕРАТУРА

1. Ганеш К., Девараж Д. Виявлення вторгнень за допомогою штучної нейронної мережі зі зменшеними вхідними можливостями. URL: http://ictactjournals.in/paper/IJSC_V1_I1_PAPER_5_30_36.pdf. Коїмбатор, 2010. 36 с.
2. Кумар Г., Кумар К., Сачдева М. Використання методів на основі штучного інтелекту для виявлення вторгнень. Пенджаб: Springer Science + Business, 2010. 387 с.
3. Гупта Б., Сінг Б., Джейн В. Виявлення вторгнення за дпомогою штучного інтелекту. URL: https://arxiv.org/ftp/arxiv/papers/1701/1701.02145.pdf. Нью Делі, 2014. 43 с.

4. Сікос Л. Ф. Штучний інтелект у кібербезпеці. Нью Йорк: Springer, 2018. 205 с.
5. Документація з офіційної сторінки «Wireshark»: веб сайт. URL: https://www.wireshark.org/docs/.
6. Тійгу Є. Штучний інтелект у кіберзахисті. Таллінн, 2011. 105 с. URL: https://www.ccdcoe.org/uploads/2018/10/ArtificialIntelligenceInCyberDefense-Tyugu.pdf.
7. Сараті Бхаттачарджі П., Ара Бегум С. Нечіткий підхід для системи виявлення вторгнень. Агаратала: Міжнародний журнал досліджень у галузі комп'ютерних наук та техніки (IJRSCSE), 2013. 108 с.
8. Шахріар Усман Хан, Фаріха Еусуфзай, Мад. Азхаруддін Редван, Мохіуддін Ахмед, Сайфур Рахман Сабудж. Штучний інтелект для кібербезпеки: аналіз продуктивності мережі по виявленню вторгнень. Кам: Springer, 2022. 140 с.

REFERENCES

1. Ganesh Kumar P., Devaraj D. Intrusion detection using artificial neural network with reduced input features. URL: http://ictactjournals.in/paper/IJSC_V1_I1_PAPER_5_30_36.pdf. Coimbatore, 2010. 36 p.
2. Kumar G., Kumar K., Sachdeva M. The use of artificial intelligence-based techniques for intrusion detection. Punjab: Springer Science + Business, 2010. 387 p.
3. Gupta B., Singh B., Jain V. Artificial Intrusion Detection Techniques. URL: https://arxiv.org/ftp/arxiv/papers/1701/1701.02145.pdf. New Delhi, 2014. 43 p.
4. Sikos L. F. AI in Cybersecurity. New York: Springer, 2018. 205 p.
5. Documentation from official page of «Wireshark»: web site. URL: https://www.wireshark.org/docs/.
6. Tyugu E. Artificial Intelligence in Cyber Defense. Tallin, 2011. 105 p. URL: https://www.ccdcoe.org/uploads/2018/10/ArtificialIntelligenceInCyberDefense-Tyugu.pdf.
7. Sarathi Bhattacharjee P., Ara Begum S. Fuzzy Approach for Intrusion Detection System. Agartala: International Journal of Research Studies in Computer Science and Engineering (IJRSCSE), 2013. 108 p.
8. Shahriar Usman Khan, Fariha Eusufzai, Md. Azharuddin Redwan, Mohiuddin Ahmed, Saifur Rahman Sabuj. Artificial Intelligence for Cyber Security: Performance Analysis of Network Intrusion Detection. Cham: Springer, 2022. 140 p.

Надійшла у першій редакції 16.01.2022, в останній - 20.04.2022.