УДК 681.04 DOI: 10.26565/2519-2310-2022-1-01

ВИКОРИСТАННЯ НЕЙРОННОЇ МЕРЕЖІ ЗАМІСТЬ БАЗИ ЗНАНЬ У ЕКСПЕРТНІЙ СИСТЕМІ ДЕТЕКТОРУ ЗЛОВМИСНОГО ТРАФІКУ ДО ВЕБ-РЕСУРСІВ

Поліна Рогоза, Віталій Єсін

Харківський національний університет імені В.Н. Каразіна, майдан Свободи, 4, Харків, 61022, Україна
polina.rohoza@gmail.com, v.i.yesin@karazin.ua

Надійшла: червень 2022. Прийнята: липень 2022.

Анотація: *Сучасний світ інформаційних технологій надає нам широкий спектр веб-застосунків. Звісно, існує постійна необхідність у надійному захисті веб-ресурсів та конфіденційної інформації, яка на них зберігається. Зі збільшенням числа кібератак зростають також критичні наслідки від них не тільки для приватних осіб, але і для підприємств. В роботі розглянуто елементи експертної системи та здійснено оцінювання їх ефективності. Основна мета застосування експертної системи – підвищення захищеності веб-ресурсів від кібератак (типу SQLi, XSS, SSI, BufferOverflow тощо) шляхом забезпечення швидкої обізнаності фахівців інформаційної безпеки про наявність атаки. Нейронна мережа здатна детектувати та класифікувати зловмисний трафік веб-серверів. До переваг застосування нейронної мережі відносяться: ефективна побудова нелінійних залежностей, адаптація до змін та оцінювання атак "нульового дня", відмовостійкість, відносна простота реалізації, швидкість обчислення після навчання. Результатом роботи є розроблений елемент експертної системи – навчена та верифікована модель нейронної мережі, яка гарантує 98% успішності детектування кібератак на веб-ресурси, а також менше 5% виникнення помилок першого та другого роду у відповідності до використаного набору даних.*

Ключові слова: *експертна система; нейронна мережа; захист веб-додатків; кібератака.*

1. Вступ

Підвищення захищеності веб-ресурсів є однією з найважливіших сфер інформаційної безпеки (ІБ). Збільшення кількості веб-ресурсів неминуче веде за собою до збільшення числа кібератак, погіршуються наслідки від цих атак них не тільки для приватних осіб, але і для підприємств. Проблема постійного збільшення чисельності кібератак потребує своєчасного інформування фахівців ІБ про поточний стан кіберзагроз. На сучасному етапі технологічного розвитку є кілька підходів для вирішення зазначеної проблеми, зокрема за рахунок впровадження автоматичних інтелектуальних систем. Однак, нині засоби захисту, які засновані на штучних нейронних мережах, досі надають широкий потенціал для різних наукових випробувань, зокрема для захисту веб-ресурсів, і не є повноцінно дослідженими. Враховуючи те, що застосування технології нейронних мереж (НМ) надає багато практичних можливостей і переваг, безумовно, дана тематика є вкрай актуальною.

В межах вирішення зазначеної проблематики в даній роботі розглядається можливість використання експертної системи (ЕС), що побудована на основі технологій НМ. Дана ЕС дозволяє детектувати зловмисний трафік веб-серверів. Створення автономної ЕС дозволяє знизити навантаження на фахівців ІБ, підвищити швидкість аналізу трафіку та виключити «людський фактор». В цілому це дозволить зменшити втрати підприємств (економічні, інформаційні, репутаційні) та підвищити рівень захищеності веб-застосунків від кібератак.

2. Основні поняття та пов'язані роботи

Експертна система – це комп’ютерна система штучного інтелекту, призначена для вирішення складних проблем (*прогнозування, контролювання, управління тощо*) і надання можливості приймати такі рішення, які здатна прийняти і людина-експерт. ЕС виконує це, беручи знання зі своєї бази знань, використовуючи правила міркувань і висновків відповідно

до запитів користувачів. Продуктивність ЕС базується на знаннях експерта, які зберігаються в базі знань.

Експертна система в основному складається з трьох компонентів:

- інтерфейс користувача (*англ. User Interface*);
- машина логічного виведення (*англ. Interface Engine*);
- база знань (*англ. Knowledge Base*).

Нейронна мережа (модель) – це метод штучного інтелекту, у якому комп'ютерні системи обробляють, аналізують, класифікують дані подібно спрощеній нервовій системі мозку людини. НМ працює завдяки великої кількості взаємопов'язаних абстрактних нейронів. Вони є блоками обробки даних, розташованими пошарово у відповідності до конкретної архітектури. У класичному варіанті архітектури завжди присутні вхідний шар даних, прихований шар математичних перетворень даних та вихідний шар.

Застосування НМ здійснюється поетапно:

1. Постановка завдання – формування мети застосування НМ.
2. Навчання НМ – послідовний процес зміни синаптичних ваг у початковій моделі, що відображають силу збудження зв'язків між нейронами. Початкова модель НМ, як правило, виконує велику кількість контрольованих навчальних завдань, будуючи стратегію прийняття рішень з того набору даних, де правильна відповідь надається заздалегідь. За кожне таке завдання початкова модель налаштовує показники ваг зв'язків між нейронами, підвищуючи точність своїх прогнозів. Для налаштування модель порівнює початкові результати з наданою правильною відповіддю або цільовим показником та фіксує певні кореляції. Цей процес НМ повторює багато разів, прагнучи покращити прогнозуючу здатність при налагодженні моделі. У результаті відбувається еволюція початкової моделі у навчену НМ.
3. Експлуатація НМ – пред'явлення нейронній моделі деяких нових невідомих ситуацій, які або розпізнаються або ні.

Нейронні мережі мають наступні переваги:

– Універсальність. Велика частина НМ здатна аналізувати навіть недосконалі вхідні дані – неповні, недостатні, занадто комплексні, які включають велику кількість параметрів. На всю вихідну генерацію не впливає пошкодження одного чи кількох нейронів, що робить нейронні мережі водночас відмовостійкими.
– Відносна простота. Побудова НМ за допомогою сучасних програмних засобів не є складною та не займає багато часу. Окрім цього, розробнику не обов'язково повноцінно розуміти внутрішнє функціонування штучних нейронів.
– Вихідні дані або результати НМ не обмежуються кількістю вхідних даних.

Однак при використанні НМ необхідно враховувати і певні недоліки, властиві їм:

– Проблема вибору архітектури мережі. На відміну від необов'язковості знання про те, як функціонує НМ, розробнику необхідно успішно підібрати архітектуру шарів (*яких існує велика кількість*) для певного завдання. До того ж, стандартних архітектурних рішень може не існувати.
– Водночас з властивістю відносної простоти, існує також проблема інтерпретації результатів роботи, оскільки складні внутрішні механізми НМ залишаються «*чорним ящиком*». У ситуації, коли необхідно пояснити, на чому ґрунтуються передбачення моделі, часто це зробити неможливо.

Сучасні нейронні мережі продовжують відмінно впорюватись з проблемами класифікації, прогнозування, кодування і декодування інформації, а також мінімізують суб'єктивну та

упереджену складову оцінки, що задовольняє поточній задачі детектування (*прогнозування та класифікації*) кіберзагроз у *HTTP*-трафіку. Окрім цього, підходи машинного навчання до програм кібербезпеки пропонують розумну можливість ідентифікувати нові модифікації зловмисного програмного забезпечення та атак «нульового дня», що особливо важливо у стрімкому темпі розвитку нових технології.

Розглянемо більш детально, існуючи релевантні нейронні моделі (НМ) і методи для обраної предметної області (*кібербезпека*). Наприклад, в роботі [1] група авторів (*Корченко та ін.*) запропонувала підхід до виявлення кібератак та вразливостей інформаційно-телекомунікаційних систем, згідно з яким розпізнавання атак за допомогою НМ зводиться до оцінок величин параметрів безпеки ресурсів інформаційної системи. Так, якщо визначена за допомогою НМ оцінка перевищує певне граничне значення, то вважається, що кібератака виявлена. У протилежному випадку вважається, що рівень безпеки знаходиться в допустимих межах. В межах цієї роботи дослідниками були представлені, описані і проаналізовані (*в порівнянні один з одним*), наступні нейромережеві методи та моделі [1]:

1) Методи простої та семантичної класифікації мережевих атак;
2) Нейромережевий підхід виявлення SQL-ін’єкцій;
3) Бінарний нейромережевий метод;
4) Метод використання НМ гібридної структури типу *Counter Propagation*;
5) Адаптивна система виявлення мережевих атак;
6) Метод побудови сукупного класифікатора трафіку тощо.

У дослідженні [2] автори пропонують обчислювальну систему на основі інтелектуальних гібридних моделей, яка за допомогою нечітких правил дозволяє будувати експертні системи в домені класифікації кібернетичних вторгнень, зосереджуючись на атаці *SQL* Injection.

Автори роботи [3] зазначають, що останнім часом для виявлення кіберзагроз, таких як мережева атака, проникнення шкідливого програмного забезпечення або фішинговий вебсайт, використовувалося кілька моделей глибокого навчання. При цьому останні зазвичай страждають від того, що не можуть бути пояснені експертами з безпеки. Експертам з безпеки необхідно не тільки виявляти вхідні загрози, але й знати вбудовані функції, які спричиняють цей конкретний інцидент безпеки. Тому *Mahdavifar* та інші [3] пропонують свою експертну систему глибокої вбудованої нейронної мережі (*deep embedded neural network expert system, DeNNeS*), яка отримує уточнені правила з архітектури навченої глибокої нейронної мережі (*deep neural network, DNN*) для заміни бази знань експертної системи. Пізніше база знань використовується для класифікації невидимого інциденту безпеки та інформування кінцевого користувача про відповідне правило, яке зробило цей висновок.

Все це свідчить про зацікавленість наукової спільноти у розвитку нейронних мереж у домені ІБ та проведення додаткових досліджень для вирішення наявних проблем.

3. Аналіз рейтингів OWASP та CWE

Визначимо релевантні кібератаки на веб-застосунки згідно рейтингів *OWASP* та *CWE Top 10*. Рейтинг *OWASP Top 10* 2017-2021 років, представлений на рис. 1, з якого можна побачити основні тренди зміни загроз ІБ за минулі 4 роки [4].

Актуальний рейтинг найбільш небезпечних загроз *CWE* наведений у таблиці 1 [5].

Відзначимо, що нейронна мережа отримує на вхід, та зможе розпізнавати шкідливий трафік, тобто вхідні дані (запити) до веб-серверів. У даному випадку аналізу піддаються усі атаки типу ін’єкцій, які використовують вхідні данні у формі модифікованого *HTTP* або *URL*-запиту, які посилають до веб-серверу.

Отже, після розгляду рейтингів *OWASP Top 10* та *CWE Top 10*, визначимо набір актуальних та додаткових атак, які експертна система має бути здатна розпізнавати [6]:

- *SQL Injection;*
- *Cross Site Scripting (XSS);*
- *XML External Entities Injection (XEE);*
- *Server-Side Includes Injection (SSI);*
- *Buffer Overflow;*
- *Carriage Return Line Feed Injection (CRLFi);*
- *XPath;*
- *Lightweight Directory Access Protocol Injection (LDAPi);*
- *Format String*.

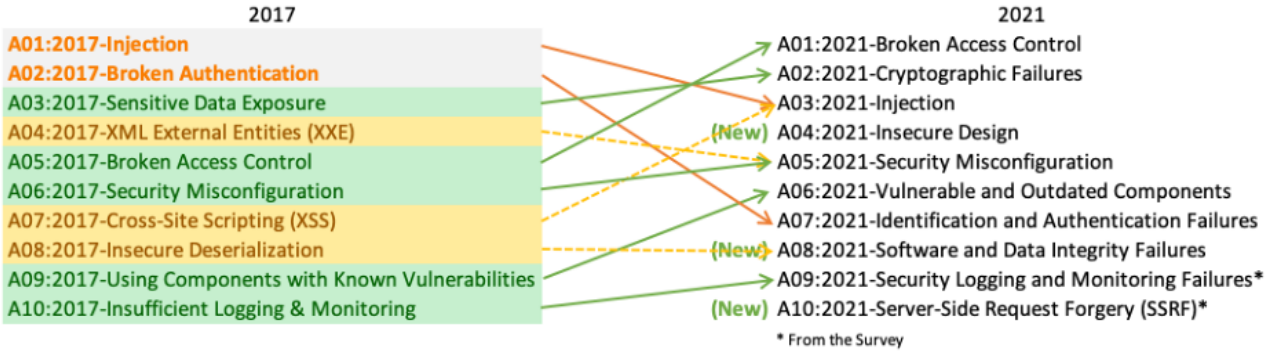

Рис. 1 – Рейтинг *OWASP Top 10 2021*

Таблиця 1 – Рейтинг загроз *CWE 2022*

№	Позначення	Назва	Оцінка
1	CWE-787	Запис даних поза меж (*Out-of-bounds*)	64.20
2	CWE-79	Неправильна нейтралізація вводу під час генерації веб-сторінок (Міжсайтовий скріптинг, *XSS*)	45.97
3	CWE-89	Неправильна нейтралізація спеціальних елементів при використанні команд SQL (Ін'єкція *SQL*)	22.11
4	CWE-20	Неправильна перевірка вводу	20.63
5	CWE-125	Читання даних поза меж (*Out-of-bounds*)	17.67
6	CWE-78	Неправильна нейтралізація спеціальних опцій при використанні команд ОС (Ін'єкція командного рядка ОС)	17.53
7	CWE-416	Використання динамічної пам'яті після її звільнення (*Use-After-Free*)	15.50
8	CWE-22	Неправильне обмеження шляху до каталогу з обмеженим доступом *(Path Traversal)*	14.08
9	CWE-352	Міжсайтова підробка запиту (*CSRF*)	11.53
10	CWE-434	Необмежене завантаження файлу небезпечного типу	9.56

4. Проектування та розробка нейронної мережі

При розробці НМ, перш за все, необхідно обрати придатну архітектуру моделі для задачі прогнозування і класифікації обраних класів кібератак. Існує багато відомих моделей класифікаторів, наприклад, метод k-найближчих сусідів (*K-NN)*, класифікатори *Naive Bayes, Random Forest, Decisions Tree*, НМ без нагляду (*вчителя*).

Зазначимо, що вхідний запит може або завдавати шкоди веб-ресурсу шляхом ін'єкції, або бути звичайним у рамках його трафіку. У цьому випадку класифікація містить результати «звичайний запит» (*normal*), або «зловмисний запит».

Проте, у даному випадку буде проводитись не бінарна, а багатокласова класифікація, адже, як показав аналіз рейтингів, ін'єкційні атаки можуть мати різні форми: деякі типи *SQL*-ін'єкцій, впровадження *XML, JSON* або *JS*-коду в запит, БД, *HTML*-код сторінки. Окрім цього, також виділимо клас *anomalous*, який відображає атаку «нульового дня». Для вирішення даної задачі класифікації пропонується застосувати згорткову нейронну мережу [6].

Згорткові нейронні мережі (ЗНМ, *англ. convolutional neural network, CNN,*) спочатку були розроблені для класифікації об'єктів зображень, а згодом вони успішно поширитись на обробку природної мови та текстових даних.

Дослідження наукової області розробок допомагають узагальнити процес побудови багатошарової нейронної мережі та визначити низку таких необхідних етапів [7].

1) Визначити вектор вхідних даних. Він повинен містити повністю всю достатню та необхідну для подальшого прогнозування інформацію.
2) У повній мірі визначити всі складові вихідного вектору, необхідні для повноцінної відповіді для поставленої задачі.
3) Вибрати оптимальну з точки зору завдання функцію активації нейронів.
4) Визначити архітектуру: тип та кількість шарів, і число нейронів у кожному з них.
5) Присвоїти початкові значення ваг і порогових рівнів. Для збереження прийнятної швидкості навчання, значення не повинні бути як великими, так і малими.
6) Провести навчання найкращим можливим чином, тобто вдало підібрати функцію втрат, за необхідності – оптимізатор, кількість епох навчання. Це налаштування дозволить уникнути повільного навчання і, у подальшому, перенавчання. У результаті НМ зможе вирішувати подібні невідомі завдання.
7) Перевірити успішність функціонування мережі шляхом подання на її вхід задачі у вигляді знайомого вхідного вектору. Далі оцінити наданий моделлю результат рішення у вигляді вихідного вектору на предмет справжності та дійсності.

Для програмної розробки нейронної мережі використовуються інструменти: *Jupyter Notebook* (*IDE з відкритим кодом для інтерактивної розробки та представлення наукових проектів*), мова програмування *Python 3.8*, бібліотеки *TensolFlow, Keras, sklearn, pandas*.

У цьому дослідженні навчання моделі проводиться на публічному наборі даних *CSIC 2012 Dataset*. Він був штучно сформований за допомогою фреймворку *Torpeda* [8] для веб-сервісу електронної комерції та навмисно містить суттєву кількість зловмисного трафіку для тестування засобів захисту.

CSIC 2012 Dataset містить усього 74 133 запитів, з яких: 49 311 є зловмисними, з різним типом ін'єкцій, 16 459 є аномальними (клас «*anomalous*» у програмній реалізації) та 8 363 є звичайними (клас «*normal*» у програмній реалізації). Кількість запитів для ін'єкційних класів наступна [6]:

43 013 – різновиди *SQL Injection*;
4818 – різновиди *XSS*;
451 – *SSI* (*Server-Side Includes Injection*);
412 – *Buffer Overflow*;
327 – CRLF (*Carriage Return Line Feed*);
175 – *XPath*;
74 – *LDAPi* (*Lightweight Directory Access Protocol Injection*);
41 – *Format String*.

На рис. 2 можна побачити програмну реалізацію поділу набору даних за класами.

Dataset був об'єднаний у *Excel* таблицю *(тип файлу csv)*, кожен рядок якої підлягає виразу [6]:

- file (*назва вихідного файлу*), id (*номер запиту у певному класі*);
- label (відмітка класу);
- method (*методи HTTP-протоколу – GET або POST*);
- path (*частина URL, шлях до сторінки, яка відображається клієнту*);
- query (*набір параметрів, що передаються у запиті*);
- url (*повне посилання – path + query*).

Також *dataset* необхідно поділити на дві окремі частини у відношенні: данні для навчання моделі (70%) і данні для її тестування (30%).

```
df.to_csv('./all.csv')
```

```
df['label'].value_counts()
```

```
SQLi              43013
anomalous         16459
normal             8363
XSS                4818
SSI                 451
BufferOverflow      412
CRLFi               327
XPath               175
LDAPi                74
FormatString         41
Name: label, dtype: int64
```

Рис. 2 – Класифікація запитів набора даних

Наступним кроком буде проведення маніпуляцій з частиною набору даних для навчання, які носять назву обробки природної мови (*Natural Language Processing, NLP*). Мета попередньої обробки (*preprocessing*) полягає у переведенні текстових символів у формат, з яким у подальшому буде працювати НМ. Клас ***keras.preprocessing.text.Tokenizer*** є реалізацією методів векторизації текстових даних, тобто методів репрезентації тексту у числовому поданні. Таким чином, слова або фрази з кожного вхідного текстового значення відображаються у відповідний вектор дійсних чисел зі спільного для усіх запитів словника (*рис. 3*). Модель застосовує поточний вектор для пошуку передбачень на основі його подібності до інших векторів з відомим результатом.

```
# Character-level word segmentation, fitting on the training set
tokenizer = Tokenizer(filters='\t\n', char_level=True)
tokenizer.fit_on_texts(url_train)
# Build a dictionary and save
num_words = len(tokenizer.word_index)+1
vocab = tokenizer.word_index
print("The size of the dictionary is %d" % num_words)
print("dictionary: ")
print(vocab)
with open("./tokenizer/vocab.json", 'w') as f:
    json.dump(vocab, f, ensure_ascii=False)
```

```
The size of the dictionary is 72
dictionary:
{'%': 1, '2': 2, 'c': 3, '0': 4, 'i': 5, 'e': 6, 'r': 7, 'o': 8, '3': 9, 'a': 1
0, 'n': 11, '=': 12, 'l': 13, 'm': 14, '&': 15, 'd': 16, '1': 17, '7': 18, 't':
19, 's': 20, 'p': 21, '5': 22, '6': 23, 'u': 24, '9': 25, '8': 26, 'b': 27,
'/': 28, ',': 29, '4': 30, 'g': 31, '.': 32, 'f': 33, 'j': 34, 'w': 35, ' ': 3
6, '?': 37, 'v': 38, 'h': 39, 'z': 40, 'x': 41, '+': 42, 'y': 43, '-': 44, 'k':
45, '#': 46, 'q': 47, ';': 48, '<': 49, '>': 50, '"': 51, '_': 52, '@': 53,
'*': 54, ':': 55, '(': 56, ')': 57, "'": 58, '!': 59, '[': 60, ']': 61, '{': 6
2, '}': 63, '`': 64, '\r': 65, '$': 66, '~': 67, '|': 68, '\\': 69, '^': 70,
'\n': 71}
```

Рис. 3 – Словник для векторизації запитів

Архітектура кожної згорткової нейронної мережі обов'язково включає в себе наступні структурні шари у відповідній послідовності: згорткові шари (англ. *Convolutional layer*), пулінгові шари (англ. *Pooling layer*) і повнозв'язні шари (англ. *Fully-connected layer*). Дана програмна реалізація заснована на Sequential-моделі модулю *Keras*, яка містить у собі усі необ-

хідні шари з певною конфігурацією (*див. рис. 4*). Нижче наведений короткий опис сенсу введення кожного окремого шару моделі [6].

Шар *Embedded* можна представити у вигляді простого матричного множення, яке перетворює натуральні числа (індекси) на щільні вектори фіксованого розміру. *Conv1D* – це шар, який створює згорткове ядро за одним просторовим (або часовим) вимірюванням. Шар *Pooling* (Об'єднання) зменшує кількість параметрів, для подальшого навчання та, відповідно, обсяг обчислень мережі. Рівень об'єднання зберігає тільки найбільш суттєві ознаки, згенеровані попередньо шаром згортки.

```
# Build a network
model = Sequential()
model.add(layers.Embedding(num_words, 64, input_length=max_len))
model.add(layers.Conv1D(32, 7, activation='relu'))
model.add(layers.MaxPooling1D(5))
model.add(layers.Conv1D(32, 7, activation='relu'))
model.add(layers.GlobalMaxPooling1D())
model.add(layers.Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())
```

Рис. 4 – Словник для векторизації запитів

Щільний шар (*Dense*) є повністю пов'язаним із попереднім шаром. Нейрони щільного шару здійснюють матрично-векторне множення. *Dense* вводиться наприкінці, для зміни розмірності виходу (у даному випадку для 10 класів кінцева розмірність шару має бути 10) та реалізує операцію *output = activation (dot (input, kernel) + bias).*

Функція активації (*activation*) за певним правилом визначає, чи є внесок окремого нейрону в мережу значним і чи потрібно його зберегти надалі.

Як можна побачити на рис. 4, були застосовані дві найпопулярніші функцій активації.

Rectified Linear Unit (*ReLU*) – це нелінійна функція активації, яка повертає 0 у разі негативного аргументу, а при позитивному аргументі – його чисельне значення залишається незмінним на виході функції (*див. рис. 5*) [9].

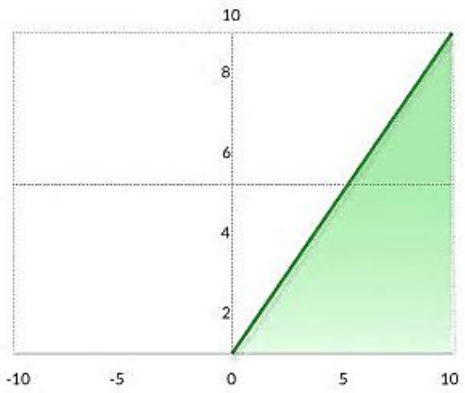

Рис. 5 – Графік функцій *ReLU*

Функція *Softmax* – це узагальнення логістичної функції для багатовимірного випадку. Вона застосовується для задач класифікації, коли кількість можливих класів більше двох.

Функція втрат – це міра того, наскільки добре модель прогнозування передбачає очікуваний результат (*або значення*). Бажано звести її до мінімуму. Для задач машинного навчання в переважній більшості випадків використовується крос-ентропія (*тобто, перехресна ентропія*), яка регулює ваги нейронів НМ під час тренування [10].

Оптимізатор – це алгоритм, який вирішує поширену проблему надто повільного навчання моделі. Він дозволяє зменшити час та скоригувати якість навчання моделі також шляхом певної незначної зміни ваг нейронів. Пропонується використовувати оптимізатор *Adam* – це один з найбільш ефективних алгоритмів оптимізації, який можна використовувати замість класичної процедури стохастичного градієнтного спуску для ітеративного поновлення ваг мережі на основі навчальних даних [11].

На рис. 6 представлений програмний звіт про спроектовану архітектуру моделі, кількість нейронів на кожному шарі та кількість вхідних параметрів. Навчання моделі відбувалося на ноутбуку Asus з процесором Intel *Core i5-6198DU CPU 2.3GHz* [6]. Процес складався з шістьох послідовних ітерацій (*тобто епох*), що емпірично визначено як оптимальна кількість, щоб уникнути як проблем недостатнього навчання, так і перенавчання. Кожна епоха тривала приблизно 5 сек. та наприкінці включала перевірку якості на контрольній множині. Завдяки цьому, у кожній наступній ітерації навчання відбувається на відкоригованих вагах і, як наслідок, зменшується величина функції втрат та підвищується точність передбачення. Після навчання НМ помилки детектування кібератак становлять 1,38% і точність – 99,56%.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 600, 64)           4608
_________________________________________________________________
conv1d (Conv1D)              (None, 594, 32)           14368
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 118, 32)           0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 112, 32)           7200
_________________________________________________________________
global_max_pooling1d (Global (None, 32)                0
_________________________________________________________________
dense (Dense)                (None, 10)                330
=================================================================
Total params: 26,506
Trainable params: 26,506
Non-trainable params: 0
_________________________________________________________________
```

Рис. 6 – Звіт про архітектуру моделі

5. Верифікація нейронної моделі

Для тестування отриманих показників точності НМ використаємо другу частину попередньо розділеного набору даних. Підготовка до верифікації включала заходи: – усі запити були розміщені у файлі *url_test.txt*, а відомі відповідні до них мітки класів – у файлі *label_test.txt*; – на вхід нейронної моделі подавався *url_test.txt*, а результат детектування кібератак (моделлю) ставився у відповідність до *label_test.txt*, тобто НМ не мала доступу до дійсних міток класу на цей раз; – вручну проводилось порівняння фактичних показників із передбаченими. Результат верифікації показує, що нейронна мережа стовідсотково впоралась з передбаченням перших десяти запитів (*див. рис. 7*).

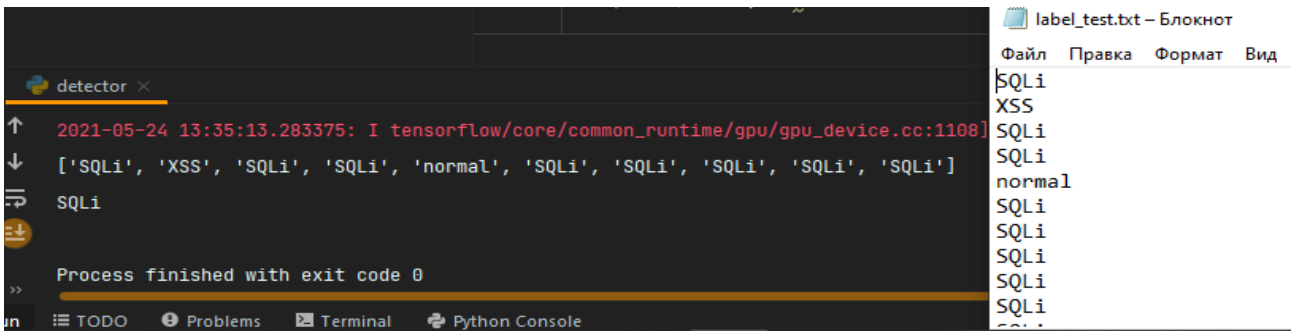

Рис. 7 – Результат виконання програми

В галузі машинного навчання, й зокрема для проблеми класифікації, матриця невідповідностей (англ. *confusion matrix*), або матриця помилок (англ. *error matrix*), - це метод підсумовування продуктивності алгоритму класифікації. Дана матриця відображає порівняне про-

гнозоване значення з фактичним значенням (рис. 8). Кожен рядок цієї таблиці є екземпляром фактичного класу змінної, тоді як кожен стовбець, є екземпляром прогнозованого класу.

З матриці помилок видно, що НМ є дійсно продуктивною та помилки першого та другого роду майже для усіх класів не виникають (*перетин стовпця та рядка дорівнює 0*).

На рис. 9 представлені метрики для кожного класу атак зокрема і загальна точність моделі. Як висновок, у контексті дослідження на наборі даних *Torpeda*, модель *CNN* є достатньо ефективною та детектує переважно всі ін'єкції у *HTTP*-запитах.

```
[[12903     0     0     0     0     0     0     1     0     0]
 [    3  4919    16     0     0     0     0     0     0     0]
 [    0    17  2492     0     0     0     0     0     0     0]
 [    6     0     0  1437     0     0     0     1     0     1]
 [    0     6     3     0   125     0     0     0     0     1]
 [    0     0     0     0     2   122     0     0     0     0]
 [    0     0     0     0     0     0    98     0     0     0]
 [   31     0     0     2     4     0     0    15     0     1]
 [    0     0     0     0     0     0     0     0    21     1]
 [    0     1     0     0     0     0     0     0     0    11]]
```

Рис. 8 – Матриця невідповідностей (*помилок*)

	precision	recall	f1-score	support
SQLi	1.00	1.00	1.00	12904
anomalous	1.00	1.00	1.00	4938
normal	0.99	0.99	0.99	2509
XSS	1.00	0.99	1.00	1445
SSI	0.95	0.93	0.94	135
BufferOverflow	1.00	0.98	0.99	124
CRLFi	1.00	1.00	1.00	98
XPath	0.88	0.28	0.43	53
LDAPi	1.00	0.95	0.98	22
FormatString	0.73	0.92	0.81	12
accuracy			1.00	22240
macro avg	0.96	0.90	0.91	22240
weighted avg	1.00	1.00	1.00	22240

Рис. 9 – Метрики якості НМ

5. Висновки

1. У запропонованій роботі розглянуті релевантні загрози ІБ у рейтингах *OWASP* та *CWE*, і на основі їх аналізу був сформований перелік атак типу ін'єкцій для детектування.

2. В результаті проведення досліджень:

– продемонстровано, що застосована архітектура згорткової нейронної мережі є доцільною і оптимальною для розподілу HTTP-запитів за класами *anomalous, normal, SQLi, XSS, SSI, BufferOverflow, CRLFi, XPath, LDAPi, FormatString*;

– проведено успішне навчання моделі на наборі даних із 74 133 унікальних запитів;

– верифікація показала, що *CNN* гарантує 98% успішності детектування кібератак на веб-ресурси, а також менше 5% виникнення помилок першого та другого роду у відповідності до використаного набору даних *Torpeda*.

3. Спроектована модель НМ довела свою продуктивність та доцільність. Враховуючи такі переваги, як простота, швидкість побудови, незначне використання технічних ресурсів, вона може бути застосована при плануванні та реалізації відповідної експертної системи.

4. В умовах реального мережевого трафіку точність НМ буде нижче через можливу первинну непристосованість. Тому у подальшому для відстеження еволюції у перенавчанні та вдосконалення використовуваної моделі, доцільно реалізувати відповідну базу даних.

Список літератури

[1] Корченко, О. Г., Терейковський, І. А., Дзюбаненко, А. В. (2014). Сучасні нейромережеві методи та моделі оцінки параметрів безпеки ресурсів інформаційної системи. Вилучено із https://doi.org/10.18372/2410-7840.16.7539

[2] Batista, L. O., de Silva, G. A., Araujo, V. S., Araujo, V. J. S., Rezende, T. S., Guimarães, A. J., Souza, P. V. D. C. (2019). Fuzzy neural networks to create an expert system for detecting attacks by sql injection. Вилучено із https://doi.org/10.48550/arXiv.1901.02868

[3] Mahdavifar, S., Ghorbani, A. A. (2020). DeNNeS: deep embedded neural network expert system for detecting cyber attacks. Neural Computing and Applications. Вилучено із https://doi.org/10.1007/s00521-020-04830-w

[4] OWASP Top 10 Application Security Risks. (2021). Вилучено із https://owasp.org/Top10/

[5] Common Weakness Enumeration (CWE) Top 25 Most Dangerous Software Weaknesses. (2022). Вилучено із https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25.html

[6] Рогоза П. В. Оцінка захисту web-ресурсів від кібератак. Пояснювальна записка до дипломної роботи бакалавра: напрям підготовки 125 – Кібербезпека / П. В. Рогоза; Харківський національний університет імені В. Н. Каразіна. – Харків: [Б. В.], 2021. – 67 с.

[7] Дунець, Р. Б., Рак, Ю. П., & Зачко, О. Б. (2008). Класифікація територій засобами нейронних мереж для управління проектами в забезпеченні екологічної безпеки. https://sci.ldubgd.edu.ua/jspui/handle/123456789/2505

[8] Torrano C., Perez A., Alvarez G. (2022). What is Torpeda. Вилучено із https://www.tic.itefi.csic.es/torpeda/default.html

[9] Соснин А. С., Суслова И. А. (2019). Функции активации нейросети: сигмоида, линейная, ступенчатая, RELU, TAHN. Екатеринбург: РГППУ.

[10] Гафаров Ф. М., Галимьянов А. Ф. (2018). Искусственные нейронные сети и их приложения. Уч. руководство. Казань: Издательство Казанского университета.

[11] Brownlee J. (2017). Gentle Introduction to the Adam Optimization Algorithm for Deep Learning: Deep Learning Performance. Вилучено із https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

Received: on June 2022. Accepted: on July 2022.

Authors:

Vitalii Yesin, Doctor of Engineering Sciences, Professor, Department of Security of Information Systems and Technologies, V.N. Karazin National University, Kharkiv, Ukraine.

ORCID ID https://orcid.org/0000-0003-1977-7269

E-mail: v.i.yesin@karazin.ua

Polina Rohoza, CSD Student (magistrate), Department of Security of Information Systems and Technologies, V.N. Karazin National University, Kharkiv, Ukraine.

E-mail: polina.rohoza@gmail.com

Using a neural network instead of the knowledge base in the expert system of web resources malicious traffic detector.

Abstract. The modern world of information technology provides us with a wide range of web applications. Indeed, there is a constant need for solid protection of web resources and their confidential information. As the number of cyber-attacks increases, so do their critical consequences for organizations and individuals. This work developed the elements of the expert system and evaluated their effectiveness. The main purpose of using an expert system is to increase the protection of web resources against cyberattacks (such as *SQLi, XSS, SSI, BufferOverflow*, etc.) by ensuring that information security specialists are quickly aware of the attack presence. The neural network is capable of detecting and classifying malicious web server traffic. The advantages of using a neural network include: effective construction of non-linear dependencies, adaptation to changes and evaluation of "*zero-day*" attacks, fault tolerance, relative simplicity of implementation, calculation speed after training. The result of the work is a developed element of the expert system – a trained and verified neural network model that guarantees 98% success in detecting cyberattacks on web resources, as well as errors types I and II in the neural model do not exceed 5%.

Keywords*:* Expert System; Web Application Protection; Cyber-Attack; Neural Network.