DOI: https://doi.org/10.36910/6775-2524-0560-2023-53-13
УДК 004.72
Багнюк Наталія Володимирівна, к.т.н., доцент
https://orcid.org/0000-0002-7120-5455
Лінчук Олександр Миколайович, асистент
Шипулін Олександр Олександрович, магістр
Луцький національний технічний університет, м. Луцьк, Україна

АНАЛІЗ ЛОГІВ ТРАФІКУ NGINX ЗА ДОПОМОГОЮ MACHINE LEARNING

Багнюк Н.В., Лінчук О.М., Шипулін О.О. Аналіз логів трафіку NGINX за допомогою Machine Learning. У цій статті розглядається процес використання машинного навчання для аналізу логів веб-сервера NGINX. Основна увага приділяється тому, як застосувати алгоритми машинного навчання для ефективного виявлення аномалій, шаблонів поведінки користувачів та потенційних загроз безпеці. Розглядаються підходи до збору, обробки та аналізу лог-даних, а також використання цих даних для покращення безпеки та продуктивності веб-серверів.
Ключові слова: аналіз логів, NGINX, машинне навчання, аналіз даних, виявлення аномалій, безпека веб-серверів, обробка та аналіз даних

Bahniuk N., Linchuk O., Shypulin O. Analysis of NGINX Traffic Logs Using Machine Learning. This article discusses the process of using machine learning methods for analyzing NGINX web server logs. The main focus is on how to apply machine learning algorithms for effective detection of anomalies, user behavior patterns, and potential security threats. Approaches to collecting, processing, and analyzing log data are considered, as well as using this data to improve the security and efficiency of web servers.
Keywords: log analysis, NGINX, machine learning, data analysis, anomaly detection, web server security, data processing and analysis.

Постановка задачі. В епоху цифрової трансформації, аналіз та обробка лог-даних веб-серверів, особливо в умовах постійно зростаючого трафіку, є критично важливими для забезпечення безпеки та оптимізації роботи систем. Використання алгоритмів машинного навчання може значно покращити цей процес, забезпечуючи детальний аналіз поведінки користувачів та виявлення потенційних загроз. Однак, для ефективної реалізації такого підходу важливо правильно зібрати, обробити та проаналізувати великі обсяги лог-даних, а також належним чином інтегрувати алгоритми машинного навчання у процес аналізу.

Метою дослідження є аналіз застосування методів машинного навчання для аналізу лог-даних веб-сервера NGINX. Дослідження зосереджено на визначенні ефективних підходів до збору, обробки та інтеграції цих даних з алгоритмами машинного навчання щоб забезпечити глибоке розуміння поведінки користувачів та виявлення аномалій. Дослідження надає практичні кроки та рекомендації для користувачів, які планують використовувати машинне навчання для аналізу логів своїх веб-серверів з метою покращення безпеки та продуктивності своїх систем.

Новизною дослідження є акцент на використанні методів машинного навчання для аналізу логів NGINX. Стаття детально описує процеси збору, обробки лог-даних та їх інтеграції з алгоритмами машинного навчання, надаючи практичні рекомендації для їх ефективного впровадження у робочі системи. Особлива увага приділяється аналізу типових шаблонів поведінки користувачів та виявленню потенційних аномалій, що можуть вказувати на безпекові ризики або неефективність системи.

Основна частина. Визначення ролі та значення аналізу логів NGINX у сучасних веб-системах. У контексті сучасних веб-систем, аналіз логів веб-сервера NGINX набуває особливої актуальності. Логи, які генеруються веб-серверами, є фундаментальними джерелами даних, що відображають поведінку користувачів, характеристики трафіку, а також потенційні безпекові інциденти.

У процесі аналізу логів NGINX можна виявити не тільки базові метрики, такі як обсяг трафіку чи статус відповідей сервера, але й більш складні патерни, наприклад, аномальну поведінку користувачів або спроби кібератак. Така інформація є критично важливою для забезпечення стабільності та безпеки веб-систем, а також для оптимізації ресурсів та підвищення ефективності обслуговування [1].

З огляду на вищезгадане, аналіз логів NGINX виступає як ключовий інструмент у стратегіях управління веб-інфраструктурою, дозволяючи адміністраторам та розробникам своєчасно виявляти та реагувати на проблеми, а також прогнозувати потенційні загрози. В контексті стрімкого розвитку веб-технологій та збільшення обсягів даних, компетентний аналіз логів відіграє ключову роль у підтримці високого рівня продуктивності та безпеки веб-серверів.

© Багнюк Н.В., Лінчук О.М., Шипулін О.О.

Опис основних алгоритмів машинного навчання, які використовуються для аналізу даних логів. Аналіз даних логів за допомогою методів машинного навчання відкриває нові горизонти у виявленні шаблонів, аномалій та покращенні безпекових заходів веб-систем. У цьому контексті, ключове значення набувають декілька основних алгоритмів машинного навчання [2]**.**

– Кластеризація (Clustering). Цей наглядовий метод дозволяє групувати лог-дані на основі подібності їх характеристик. Приміром, алгоритм K-means може бути використаний для ідентифікації аномальних шаблонів поведінки шляхом групування логів на кластери з типовими та нетиповими запитами.

– Класифікація (Classification). Алгоритми, такі як випадковий ліс (Random Forest) або опорні вектори (Support Vector Machines), ефективно визначають категорії лог-записів, дозволяючи швидко відокремити потенційно шкідливі або підозрілі запити від легітимних.

– Нейронні мережі (Neural Networks). Глибоке навчання через нейронні мережі забезпечує високу точність у виявленні складних шаблонів у даних. Це може бути використано для розпізнавання складних атак, як от SQL-ін'єкції, через аналіз логів.

– Аналіз часових рядів (Time Series Analysis). Застосування алгоритмів аналізу часових рядів, таких як ARIMA, дозволяє відстежувати та аналізувати тенденції у лог-даних протягом часу, що може бути корисним для виявлення поступово зростаючих загроз.

– Алгоритми виявлення аномалій (Anomaly Detection). Методи, такі як ізоляційний ліс (Isolation Forest) або однокласова SVM (One-Class SVM), виявляють відхилення від норми у поведінці системи, що допомагає швидко ідентифікувати потенційні загрози.

Увага до деталей при використанні цих алгоритмів є критичною, оскільки вірне налаштування та інтерпретація результатів є ключовими для ефективного аналізу лог-даних та виявлення потенційних загроз.

Збір та Обробка Лог-Даних NGINX. У сфері аналізу даних логів веб-сервера NGINX важливу роль відіграє розробка систем на основі методів машинного навчання, які можуть виявляти потенційно небезпечний трафік. Процес створення такої системи зазвичай складається з декількох етапів, починаючи зі збору та підготовки даних, закінчуючи розробкою та оцінкою моделі.

– Створення симульованих даних логів NGINX. Оскільки використання реальних лог-даних може бути обмеженим з різних причин, симуляція даних є важливою альтернативою. Симульовані дані можуть включати характеристики, такі як IP-адреса запитувача, дата і час запиту, HTTP-метод, URL-запиту, HTTP-статус відповіді, розмір відповіді в байтах тощо. Для створення фреймів з такими даними можна використовувати модуль pandas [4] (рис. 1).

```python
import pandas as pd

# Приклад структури симульованих даних
data = {
    "IP": ["192.168.1.1", "192.168.1.2", "192.168.1.3"],
    "Datetime": ["2023-01-08 07:26:02", "2023-01-08 07:26:05", "2023-01-08 07:26:07"],
    "Method": ["GET", "POST", "GET"],
    "Path": ["/home", "/login", "/home"],
    "Status_Code": [200, 403, 200],
    "Response_Size": [1024, 2048, 1024],
    "Is_Malicious": [0, 1, 0]
}

df = pd.DataFrame(data)
```

Рис. 1. Приклад симульованих даних логів для Nginx

– Підготовка даних. У процесі підготовки даних, ключовим завданням є перетворення сирих лог-даних у формат, придатний для аналізу машинним навчанням. Це означає, що кожна характеристика даних або «фіча» повинна бути представлена у вигляді, який може бути ефективно оброблений алгоритмами навчання. Основні етапи цього процесу включають: кодування категорійних даних та стандартизацію числових даних. Для першого може використовуватися метод One-Hot Encoding, який створює нові бінарні (0 або 1) характеристики для кожної категорії. Для другого ж використовується стандартизація, яка зміщує розподіл кожної характеристики так, щоб середнє значення було 0, а стандартне відхилення 1. Після виконання цих кроків, дані стають готовими до використання в моделях

© Багнюк Н.В., Лінчук О.М., Шипулін О.О.

машинного навчання. Ці етапи забезпечують важливу підготовку для впровадження даних у модель, допомагаючи збільшити точність та ефективність передбачень моделі (рис. 2).

```python
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split

# Розділення даних на функції та мітки
X = df.drop("Is_Malicious", axis=1)
y = df["Is_Malicious"]

# Визначення категорійних та числових колонок
categorical_columns = ['Method', 'Path']
numerical_columns = ['Response_Size']

# Створення об'єкта для попередньої обробки
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_columns),
        ('cat', OneHotEncoder(), categorical_columns)
    ])

# Попередня обробка даних
X_processed = preprocessor.fit_transform(X)

# Розділення на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X_processed, y, test_size=0.2, random_state=42)
```

Рис. 2. Підготовка даних

– Розробка моделі. Для аналізу даних логів може бути розроблена проста модель глибокого навчання за допомогою бібліотек, таких як TensorFlow [3] або PyTorch. Така модель зазвичай включає кілька щільних шарів, що дозволяє ефективно обробляти вхідні дані. Як приклад можна навести модель з 3 шарів на рисунку 3.

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Створення моделі
model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid')
])

# Компіляція моделі
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Рис. 3. 3-шарова модель для аналізу лог-даних

– Тренування та оцінка моделі. Після розробки, модель тренується на підготовлених даних. Цей процес включає в себе адаптацію ваг моделі для максимально точного прогнозування характеристик вхідних даних. Оцінка моделі здійснюється за допомогою метрик, таких як точність, щоб визначити її ефективність у класифікації логів (рис. 4).

```python
# Тренування моделі
history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.2)

# Оцінка моделі на тестовому наборі
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(f"Точність на тестовому наборі: {test_accuracy}")
```

Рис. 4. Тренування моделі та оцінка її точності

– Використання моделі. Завершальний етап полягає у використанні навченої моделі для аналізу нових лог-даних з метою ідентифікації потенційно небезпечних запитів. Після тренування та оцінки моделі на симульованих даних, її можна застосувати до нових, реальних даних логів. Нові дані логів повинні бути підготовані та оброблені відповідно до тих же процедур, що й навчальний набір. Це включає кодування категорійних даних та стандартизацію числових даних. Важливим аспектом є аналіз результатів класифікації. Для потенційно небезпечних запитів можуть бути вжиті заходи, такі як додаткова перевірка, блокування IP-адреси або сповіщення адміністраторів системи, що може бути

© Багнюк Н.В., Лінчук О.М., Шипулін О.О.

додане як опція. Ефективне використання результатів моделі може допомогти у підвищенні безпеки та стабільності веб-серверів. Приклад використання моделі наведений на рисунку 5. Під час перевірки моделі на тестовому наборі даних точність моделі була близько 50%, що можна назвати достатнім для наукового дослідження, але є очевидним те, що модель потребує більше початкових даних для того, щоб бути готовою для використання в production-сегменті.

```
(venv) → University python real_test.py
(800, 18) (200, 18)
WARNING:absl:At this time, the v2.11+ optimizer `tf.keras.optimizers.Adam` runs slowly on M1/M2 Macs, please use the legacy Keras optimizer instead, located a
t `tf.keras.optimizers.legacy.Adam`.
Epoch 1/10
20/20 [==============================] - 0s 6ms/step - loss: 0.6985 - accuracy: 0.5047 - val_loss: 0.7000 - val_accuracy: 0.4875
Epoch 2/10
20/20 [==============================] - 0s 1ms/step - loss: 0.6883 - accuracy: 0.5328 - val_loss: 0.6970 - val_accuracy: 0.5125
Epoch 3/10
20/20 [==============================] - 0s 1ms/step - loss: 0.6826 - accuracy: 0.5656 - val_loss: 0.6953 - val_accuracy: 0.4875
Epoch 4/10
20/20 [==============================] - 0s 1ms/step - loss: 0.6775 - accuracy: 0.5953 - val_loss: 0.6949 - val_accuracy: 0.4875
Epoch 5/10
20/20 [==============================] - 0s 1ms/step - loss: 0.6741 - accuracy: 0.6141 - val_loss: 0.6928 - val_accuracy: 0.5312
Epoch 6/10
20/20 [==============================] - 0s 1ms/step - loss: 0.6693 - accuracy: 0.6219 - val_loss: 0.6914 - val_accuracy: 0.5063
Epoch 7/10
20/20 [==============================] - 0s 1ms/step - loss: 0.6659 - accuracy: 0.6156 - val_loss: 0.6915 - val_accuracy: 0.5250
Epoch 8/10
20/20 [==============================] - 0s 1ms/step - loss: 0.6622 - accuracy: 0.6297 - val_loss: 0.6908 - val_accuracy: 0.5375
Epoch 9/10
20/20 [==============================] - 0s 1ms/step - loss: 0.6579 - accuracy: 0.6328 - val_loss: 0.6912 - val_accuracy: 0.5375
Epoch 10/10
20/20 [==============================] - 0s 1ms/step - loss: 0.6541 - accuracy: 0.6484 - val_loss: 0.6919 - val_accuracy: 0.5188
7/7 [==============================] - 0s 656us/step - loss: 0.7010 - accuracy: 0.5100
Точність на тестовому наборі: 0.5099999904632568
1/1 [==============================] - 0s 43ms/step
Чи є запит потенційно небезпечним: False
(venv) → University
```

Рис. 5. Приклад використання Python-скрипта для тестового запуску моделі аналізу логів

– Автоматизація за допомогою Cron. Cron – це утиліта в Unix-подібних системах, яка дозволяє запускати скрипти або команди в автоматичному режимі в заданий час Після розробки та тестування моделі необхідно вирішити питання автоматичного запуску скрипта аналізу логів на цільовій машині. Можна використати вбудований «планувальник» для автоматичного виконання за допомогою Cron. Це робиться шляхом додавання запису в crontab - файл конфігурації Cron [5]. Наприклад, щоб запускати скрипт щодня о 2 годині ночі, запис у crontab буде виглядати так, як зображено на рисунку 6.

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
0 2 * * * /usr/bin/python3 /path/to/your/script.py
```

Рис. 6. Приклад використання cron для автоматичного запуску python скрипта

– Логування. Важливо також налаштувати логування результатів виконання скрипта, щоб можна було відслідковувати його виконання та виявляти можливі помилки. Це можна зробити, наприклад, перенаправивши вивід скрипта у файл логів (рис. 7).

```
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
0 2 * * * /usr/bin/python3 /path/to/your/script.py >> /path/to/your/file.log 2>&1
```

Рис. 7. Перенаправлення текстових логів з cron-задачі в файл

© Багнюк Н.В., Лінчук О.М., Шипулін О.О.

Такий запис перенаправляє як стандартний вивід, так і помилки у вказаний файл логу.

– Тестування і Валідація: Перед тим, як залишити скрипт працювати на автоматі, важливо провести тестування, щоб переконатися, що він працює коректно і ефективно виконує свої завдання.

При аналізі відношення кількості було визначено безпосередню залежність точності моделі від кількості логів, що використовувались для її навчання на визначеному наборі параметрів (рис. 8). З діаграми, що зображена на рисунку 8, легко зрозуміти, що кількість наборів логів для навчання нижче 1000 є недостатньою для впевненості в точності моделі і навпаки, кількість наборів від 1000 і більше – позитивно впливає на точність висновків моделі.



Рис. 8. Залежність точності моделі від кількості навчальних даних

Висновки. У статті розглянуто процес розробки та застосування моделі глибокого навчання для аналізу лог-даних NGINX з метою ідентифікації потенційно небезпечного трафіку. Представлено важливі етапи, включаючи генерацію симульованих даних, підготовку даних, розробку моделі, її тренування та оцінку, а також використання моделі для аналізу нових лог-даних.

Важливим аспектом дослідження є усвідомлення того, що досягнута точність моделі хоча й відіграє позитивну роль у демонстрації можливостей глибокого навчання, не є достатньою для безпосереднього впровадження в продуктивні (продакшн) проекти. Це пов'язано з низкою чинників, включаючи обмеження симульованих даних, необхідності додаткового налаштування гіперпараметрів, а також потребу в більш ретельній перевірці моделі на різноманітних реальних даних.

Також варто підкреслити, що успішне впровадження моделей машинного навчання у сфері аналізу логів вимагає постійного моніторингу та оновлення моделі з огляду на постійно змінювані моделі поведінки користувачів та шаблони загроз.

У майбутньому дослідженні можна розглянути використання різноманітніших наборів даних для тренування та тестування моделі, а також розробку складніших архітектур нейронних мереж, які можуть більш точно моделювати складність реального мережевого трафіку. Врахування цих аспектів може значно підвищити ефективність та точність моделей у реальних сценаріях застосування.

Список бібліографічного опису

1. Overview: Metrics and Metadata | NGINX Documentation. NGINX Documentation. URL: https://docs.nginx.com/nginx-controller/analytics/metrics/overview-metrics-metadata/ (дата звернення: 02.09.2023).

2. Machine Learning Algorithms - Javatpoint. www.javatpoint.com. URL: https://www.javatpoint.com/machine-learning-algorithms (дата звернення: 05.10.2023).

3. Documentation | Terraform | HashiCorp Developer. Documentation | Terraform | HashiCorp Developer. URL: https://developer.hashicorp.com/terraform/docs (дата звернення: 12.10.2023).

4. pandas documentation – pandas 2.1.3 documentation. pandas - Python Data Analysis Library. URL: https://pandas.pydata.org/docs/ (дата звернення: 22.10.2023).

5. Crontab(5) - Linux manual page. Michael Kerrisk - man7.org. URL: https://man7.org/linux/man-pages/man5/crontab.5.html (дата звернення: 25.10.2023).

© Багнюк Н.В., Лінчук О.М., Шипулін О.О.

References

1. Overview: Metrics and Metadata | NGINX Documentation. NGINX Documentation. URL: https://docs.nginx.com/nginx-controller/analytics/metrics/overview-metrics-metadata/ (date of access: 02.09.2023).
2. Machine Learning Algorithms - Javatpoint. www.javatpoint.com. URL: https://www.javatpoint.com/machine-learning-algorithms (date of access: 05.10.2023).
3. Documentation | Terraform | HashiCorp Developer. Documentation | Terraform | HashiCorp Developer. URL: https://developer.hashicorp.com/terraform/docs (date of access: 12.10.2023).
4. pandas documentation – pandas 2.1.3 documentation. pandas - Python Data Analysis Library. URL: https://pandas.pydata.org/docs/ (date of access: 22.10.2023).
5. Crontab(5) - Linux manual page. Michael Kerrisk - man7.org. URL: https://man7.org/linux/man-pages/man5/crontab.5.html (date of access: 25.10.2023).

© Багнюк Н.В., Лінчук О.М., Шипулін О.О.