УДК 621.372(075) **DOI:10.30837/rt.2024.3.218.05**

М.С. КАВЕЦЬКИЙ, В.І. РУЖЕНЦЕВ, д-р техн. наук

ВИЯВЛЕННЯ ВЕБ-АТАК ПО HTTP ЗАПИТАМ З ВИКОРИСТАННЯМ ТЕХНІК NLP

Вступ

Зростання кількості веб-додатків і їх значущість у сучасному інтернет-просторі супроводжуються збільшенням загроз безпеці. Веб-атаки, що спрямовані на вразливості в мережевих протоколах, стають дедалі складнішими, використовуючи різноманітні техніки для обходу захисних заходів.

Основним видом атак є зловживання HTTP-трафіком, що вимагає нових та ефективних методів виявлення, оскільки старі методи мають свої слабкі місця, що призводить до хибних результатів і неправильного блокування трафіку.

Робота зосереджена на вдосконаленні методів виявлення веб-атак через аналіз HTTP-трафіку з використанням технік обробки природної мови (NLP) та моделей на базі трансформерів, зокрема BERT.

Метою дослідження є розробка високоефективного класифікатора, здатного точно ідентифікувати аномальну активність у HTTP-трафіку.

Гіпотеза, яка підтверджується в ході дослідження, полягає в тому, що рішення на основі машинного навчання можуть класифікувати зловмисний трафік з більшою точністю, ніж традиційні системи на основі правил або сигнатур, та мають нижчий рівень хибних спрацювань.

Машинне навчання. Основні поняття

Машинне навчання (ML) та обробка природної мови (NLP) є двома взаємопов'язаними областями досліджень, які значно впливають на розвиток сучасних технологій обробки даних. NLP займається взаємодією між комп'ютерами та людською мовою, тоді як ML забезпечує алгоритми та моделі, що дозволяють автоматично виявляти патерни в даних і робити прогнози на їх основі. У цьому розділі розглядаються основні поняття машинного навчання в контексті NLP, що є важливими для розуміння методів, використаних у цьому дослідженні.

Методи NLP та моделі на основі трансформерів, зокрема BERT, були використані для аналізу HTTP-трафіку та виявлення веб-атак. Трансформери, з їхньою здатністю до контекстного розуміння тексту, показали високу ефективність у задачах класифікації, що підтверджується результатами проведених експериментів. Впровадження цих методів дозволяє значно підвищити точність ідентифікації зловмисного трафіку порівняно з традиційними методами, що базуються на правилах або сигнатурах.

Токенізація є першим і основним етапом в обробці природної мови, що включає поділ тексту на окремі компоненти, звані токенами. Токени можуть бути словами, фразами або навіть окремими символами. У контексті HTTP-запитів токенізація дозволяє розділити запит на окремі частини, які можуть бути проаналізовані окремо.

Лематизація і стемінг – це техніки нормалізації тексту. Лематизація приводить слова до їх базової або словникової форми (леми), а стемінг обрізає слова до їх кореневої форми. Ці процеси допомагають зменшити кількість різновидів слів, що полегшує подальший аналіз та підвищує ефективність моделей машинного навчання.

Після токенізації та нормалізації тексту наступним кроком є векторизація, яка перетворює текстові дані в числові вектори. Одним із популярних методів векторизації є метод "мішок слів" (Bag of Words), де текст представляється як вектор кількостей слів. Більш просунуті методи включають TF-IDF (Term Frequency-Inverse Document Frequency) та векторизацію за допомогою попередньо навчених моделей, таких як Word2Vec або GloVe.

Трансформери є архітектурою глибинних нейронних мереж, що призначені для обробки послідовностей даних, таких як текст. Вони були вперше представлені в 2017 р. у статті «Attention is All You Need» [1] Вони використовують механізм самоуваги (self-attention), що дозволяє моделі фокусуватися на різних частинах вхідної послідовності для більш точного контекстного розуміння. Модель BERT (Bidirectional Encoder Representations from Transformers) є однією з найпопулярніших моделей на основі трансформерів. Вона тренується на завданнях маскування слів і прогнозування наступних речень, що дозволяє їй отримувати глибоке контекстне розуміння тексту.

Класифікація тексту є одним з основних завдань у NLP, що включає категоризацію текстових даних в один або кілька класів. У контексті виявлення веб-атак, класифікація тексту дозволяє ідентифікувати зловмисні HTTP-запити серед нормального трафіку. Для цього використовуються алгоритми машинного навчання, такі як логістична регресія, дерева рішень, випадкові ліси, градієнтний бустинг та нейронні мережі.

Отже, в цьому розділі наведено основні відомості щодо теоретичної частини, необхідної для розуміння цієї роботи. Далі буде наведено ключові переваги моделі BERT та використаної в ній архітектурі трансформерів.

Архітектура трансформерів та модель BERT

В даній роботі зосереджено увагу на використанні сучасних моделей на основі трансформерів для аналізу HTTP трафіку з метою виявлення веб-атак. Однією з ключових моделей, які розглядаються, є BERT (Bidirectional Encoder Representations from Transformers). Ця модель, створена компанією Google, стала проривом в обробці природної мови (NLP) завдяки своїй здатності розуміти контекст слів в обох напрямках, що дозволяє ефективно вирішувати різноманітні задачі, включаючи класифікацію тексту, машинний переклад та інші.

Переваги цієї моделі для задачі аналізу HTTP трафіку наступні: розуміння контексту, адаптивність до розміру вхідних даних, попереднє навчання.

Щодо першого, то вона має двонаправлений контекст: BERT аналізує текст як зліва направо, так і справа наліво, що дозволяє краще розуміти контекст і семантичні зв'язки між словами (в даному випадку слова – це текст з HTTP запиту).

Щодо адаптивності до довжини послідовностей: HTTP запити можуть варіюватися за довжиною, і BERT здатна ефективно працювати з послідовностями різної довжини.

Попереднє навчання даної моделі допомагає моделі мати велику кількість фонових знань, що допомагає їй виявляти складні загрози, які можуть бути важкими для моделей, навчальних лише на вузькому наборі даних. Також дана перевага дозволяє тренувати модель трішки швидше, бо певні базові «знання» у моделі вже є.

Дана модель має переваги з трансформерів у вигляді механізму уваги (attention mechanism), який дозволяє моделі ефективно визначати та зважувати важливість різних частин тексту відносно один одного. Це означає, що модель може зосередитися на релевантних словах і фразах, навіть якщо вони розташовані далеко один від одного в тексті, що забезпечує точне розуміння контексту. Механізм уваги працює паралельно, що дозволяє обробляти довгі послідовності тексту швидше та ефективніше, ніж традиційні рекурентні нейронні мережі. Таким чином, трансформерна архітектура робить можливим захоплення довгострокових залежностей та складних взаємозв'язків у тексті, що у випадку аналізу HTTP трафіку доволі гарно допомогає, як показують результати дослідження у наступних підрозділах.

Отже, в цьому розділі описано основні переваги щодо архітектури трансформерів, а також відомості щодо моделі BERT. У наступному розділі буде проаналізовано схожі роботи, описано основні відомості щодо реалізації детекторів атак в цих роботах, а також наведено недоліки та порівняння з запропонованим підходом.

Огляд літератури

У цій роботі була використана модель BERT зі своїми перевагами, детально описаними у попередньому розділі.

В роботі [2] також використовують датасет CSIC2010 (про датасети детально описано у наступних розділах). Але в [2] автори, на жаль, описують свої експерименти по-різному, оскільки використовують аномальний трафік в процесі генерації векторів.

Робота, яка базується на схожій ідеї, що і в цьому дослідженні, заснована на Doc2Vec методі векторизації [3]. Запити HTTP трансформуються через Doc2Vec модель в векторну форму, яка потім використовується для того, щоб передбачити чи нормальний, чи аномальний цей трафік. Ця модель натренована на повній колекції датасету CSIC2010. Вони створили ніби «документи» з 10 нормальними чи аномальними запитами, а класифікація відбувається ансамблем класифікаторів натренованих на тренувальній вибірці, яка містить 70 % всієї вибірки. І хоча стверджується, що точність класифікації наближається до 99 %, але на нашу думку, головним недоліком цієї роботи є те, що вони перевіряють свої результати на доволі обмеженому наборі даних, який може не мати багато узагальнюючих властивостей і обмежуватися тільки певними, які є в датасеті. Під час виконання цієї роботи, було зібрано великий датасет та проведено більше досліджень з-поміж моделей, створених нами, а також класифікаторів, які вже існують на ринку.

Наступна робота [4] представляє HTTP запити як біграми в словнику з 80 символів ASCII. Алгоритм Isolation Forest був використаний для визначення належності даного HTTP-запиту до нормального чи аномального трафіку в отриманому векторному просторі. Незважаючи на проведення експериментів на відомому датасеті CSIC2010, спостерігається помітна різниця в кількості даних, що використовуються, тому можна припустити, що було використано меншу підмножину даних.

Робота [5] представляє повністю керований підхід. Для класифікації HTTP-трафіку використовується нейронна архітектура LSTM-CNN. Спочатку рекурентна мережа LSTM обробляє HTTP-запит на основі блочних характеристик, потім обрані стани мережі LSTM передаються в згорткову мережу, яка після обробки векторів передає їх на вихід у вигляді MLP мережі, що класифікує HTTP-запит в один з двох класів. Автори методу повідомляють про дуже хороші результати не тільки для колекції CSIC2010, але й для колекцій CICIDS 2017 та ISCX 2012, що містять різні типи атак.

У роботі [6] описано систему виявлення веб-атак, яка базується на ансамблі класифікаторів та методах векторного подання з області обробки природної мови (NLP). Спочатку система токенізує текст на основі вручну підготовленого словника, що містить токени, характерні для мережевого трафіку. Отримані текстові представлення векторизуються паралельно за допомогою нейронних моделей, що базуються на рекурентних та згорткових мережах. Після цього проводиться комплексна перевірка, яка повертає оціночний вектор, що визначає впевненість у взаємній різниці векторів. Отриманий вектор разом з векторами з нейронних моделей передається до ансамблю класифікаторів, який оцінює, чи є HTTP-запит нормальним трафіком чи атакою. Метод був протестований на колекції CSIC2010 та власних колекціях, але, на жаль, немає інформації про те, як проводилось навчання моделі.

У роботі [7] автори представляють нову систему виявлення Locate-Then-Detect (LTD), яка може точно виявляти веб-загрози в реальному часі, використовуючи глибокі нейронні мережі з механізмом уваги. Спочатку LTD витягує підозрілі сегменти з довгих запитів, потім запропоновані області додатково перевіряються класифікатором. Завдяки ефективному фільтруванню нерелевантних рядків фону, LTD може збільшити точність на 22,3 % і зменшити 82,6 % обчислювальних витрат. Експерименти на реальному веб-трафіку показують, що LTD перевершує кілька провідних методів і може ефективно виявляти атаки із середнім часом відгуку 5 мс, що робить його добре придатним для реальних додатків. Незважаючи на численні переваги LTD, на даний момент він може обробляти лише атаки типу SQLi та XSS.

Поточна робота над LTD охоплює виявлення більшої кількості веб-атак, таких як File Inclusion і Code Execution.

У порівнянні з іншими дослідженнями, ця робота має кілька ключових вдосконалень, які роблять модель більш ефективною у виявленні веб-атак. Перша відмінність полягає у використанні сучасної архітектури трансформерів, зокрема моделі BERT, яка показала значну ефективність у задачах обробки природної мови (NLP). В певних роботах [2 – 6] не використовуються трансформери з усіма їх перевагами (описані в відповідному розділі). І хоча ці роботи використовують векторний простір для представлення датасету, проте вони базуються кожна на різних методах машинного навчання, не маючи переваг архітектури трансформерів. Робота [7], яка хоч і використовує архітектуру трансформерів, має описані вище недоліки (значна обмеженість атак, які модель може виявляти). Також серед усіх описаних в цьому розділі робіт, дана робота має ключову відмінність, яка дає більшу перевагу для моделі в області машинного навчання – більш широкий датасет. Ключова перевага побудованого нами рішення полягає в тому, що було зібрано та скомбіновано кілька датасетів, включаючи CSIC2010, PositiveTechnologies, openappsec, а також власноруч зібрані дані (про це більш детально в розділі з датасетом) для створення одного великого та збалансованого набору даних (в проаналізованих роботах датасет або використовувався менше, що дає меншу узагальнюючу спроможність для моделі машинного навчання). Датасет з цієї роботи містить понад 195 тисяч записів, що значно перевищує обсяги даних, використаних у попередніх дослідженнях. Завдяки цьому модель здатна краще узагальнювати та виявляти складні патерни в HTTP-трафіку, що забезпечує високу точність і надійність класифікації. Крім того, методи попередньої обробки даних, такі як видалення дублікативних записів і непотрібної інформації, сприяли покращенню якості моделі. В результаті, створена модель демонструє високу збалансовану точність та здатність розпізнавати складні веб-атаки з мінімальною кількістю хибних спрацювань, що робить її конкурентоспроможною серед існуючих рішень у цій сфері.

Отже, в цьому розділі було описано схожі роботи, описано їх підходи та недоліки. Певні роботи самі по собі мають недолік через використання архітектур, які не мають певних механізмів (наприклад механізму уваги), що значно послаблює їх узагальнюючі властивості під час навчання. Деякі роботи хоча і мають схожу архітектуру, проте вони будуються на інших моделях, а також на значно меншому датасету. Використання малого датасету веде до того, що модель може передбачувати менше атак (що і було продемонстровано у роботі [7]), а також ця модель сама по собі буде гірше виявляти певні атаки, бо вона не має таких переваг в узагальнюючих властивостях перед моделлю, яка навчалася на датасеті з більшою кількістю даних. В наступному розділі розглянемо особливості запропонованого підходу, а також основні відмінності між цим підходом та підходами зі схожих робіт. Також в інших розділах далі буде описано важливість вибору більш широкого датасету.

Огляд запропонованого підходу

Відомо, що методи на основі нейронних мереж вже давно застосовуються у галузі виявлення кібератак. Натомість, основна пропозиція цієї роботи полягає в удосконаленні існуючих підходів шляхом використання сучасних моделей на базі трансформерів, зокрема BERT, для аналізу HTTP-трафіку. Запропонований підхід має кілька ключових відмінностей від традиційних методів, що дозволяє досягти більш високої точності та зменшити кількість хибних спрацювань.

На рис. 1 наведено загальний підхід до класифікації. Основні відмінності між запропонованим в роботі та класичними підходами – це наявність іншого токенізатору, а також моделі (та архітектури) для класифікації.

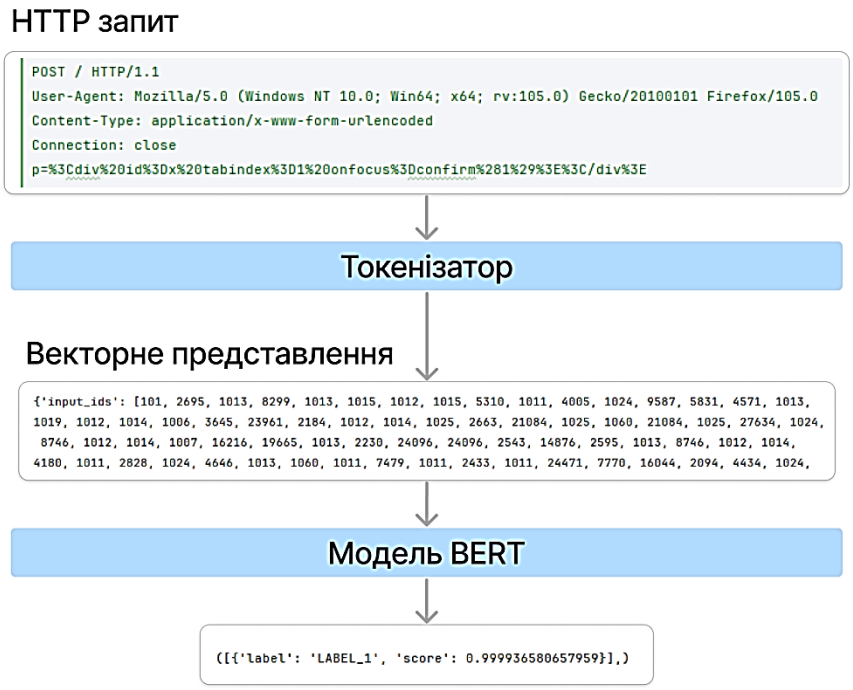

Рис. 1. Пояснення підходу

Щодо елементу токенізатору з рис. 1, то BERT токенізатор використовує WordPiece токенізацію, яка дозволяє ефективно розбивати слова на менші підслова чи символи. Це підвищує здатність моделі працювати з рідкісними або незнайомими словами, покращуючи узагальнення та розуміння тексту. Також відмінність – це подвійний процес токенізації: WordPiece токенізатор спочатку розбиває текст на токени, використовуючи регулярні вирази, а потім розбиває їх на менші підслова. Це знижує ймовірність утворення невідомих токенів (UNK) порівняно з іншими методами. BERT токенізатор є унікальним, бо враховує контекст, що дозволяє моделі краще розуміти багатозначні слова, порівняно з простішими методами токенізації, які можуть втратити контекст. Наприклад, у роботах, де використовується Doc2Vec (як у [3]), запити HTTP перетворюються в вектори без врахування глибокого контексту, що забезпечується механізмом уваги в BERT. Це може призвести до втрати важливої інформації про контекст, яку BERT зберігає. Крім того, Doc2Vec токенізатор не має тих переваг, які є у WordPiece токенізатора BERT, що дозволяє краще працювати з рідкісними або незнайомими словами. Алгоритм Isolation Forest (як у [4]) використовується для класифікації на основі просторового розподілу даних, що відрізняється від текстової обробки BERT. Він не використовує глибоке контекстне розуміння, яке забезпечує BERT, а токенізація в такій роботі зазвичай не враховує поділ на підслова, що знижує ефективність обробки рідкісних слів.

Щодо моделі (а також її архітектури) з рис. 1 можна виділити відмінності: BERT базується на архітектурі трансформерів, яка використовує механізм уваги (Attention Mechanism) для обробки всього тексту одразу, на відміну від рекурентних нейронних мереж (RNN) та LSTM, які обробляють текст послідовно. BERT навчається двонаправлено, що означає одночасне врахування контексту зліва направо і справа наліво. Це значно покращує розуміння контексту порівняно з однонаправленими моделями. А от, наприклад, в роботі [4] алгоритм Isolation Forest використовується для класифікації на основі просторового розподілу даних, що відрізняється від текстової обробки BERT. Він не використовує глибоке контекстне розуміння, яке забезпечує BERT, а токенізація в такій роботі зазвичай не враховує поділ на підслова, що знижує ефективність обробки рідкісних слів. У роботах з використанням LSTM-CNN (як у [5]), моделі обробляють послідовні дані, але можуть втратити контекст на великих відстанях у тексті, чого уникає BERT завдяки своєму глобальному механізму уваги. Токенізатори, що використовуються в цих роботах, зазвичай менш ефективні у порівнянні з

WordPiece токенізатором BERT. Роботи, які використовують ансамблі з методами NLP (як у [6]), можуть мати обмеження в узагальненні також через те, що вони не використовують трансформери, що забезпечують глибше контекстне розуміння та узагальнюючі властивості. Токенізація в таких системах часто базується на простих методах розбиття на слова або символи, без переваг WordPiece токенізації BERT.

На рис. 1 наведено пояснення щодо підходу. Можна бачити HTTP трафік, який є певним текстом і подається в токенізатор. Суть токенізатору зробити з цього тексту певні токени, які модель може розуміти. Перевага моделі BERT в тому, що для неї є натренований токенізатор, який має певний набір токенів найбільш частовикористовуваних, регістр слів не важливий, а також кожне слово – це не токен, бо одне слово може включати декілька токенів. Таким чином, після розбиття токенізатором отримуємо векторне представлення з певною довжиною. Ця довжина фіксована і під час роботи розрахована на 400 токенів, чого цілком вистачає, як буде показано далі під час проведення аналізу датасету. За потреби кількість токенів можна збільшити, проте це збільшить час, який потрібен для передбачення та тренування моделі. Далі це векторне представлення подається моделі на класифікацію, де вона видає «label» – зі значеннями «label_0», «label_1» це просто мітки класу «аномальний» або «нормальний» трафік, а також score, який показує наскільки модель впевнена в своєму передбаченні від 0 до 1. Де 1 – 100 % впевненості в тому, що саме певний label є правильним.

В машинному навчанні основні відмінності відбуваються через зміну архітектури, а також використаного датасету. Якщо в певних традиційних архітектурах був відсутній механізм уваги, то тут він є. Це дає змогу цьому методу робити більш релевантні узагальнюючі висновки щодо датасету в процесі навчання (дозволяє фокусуватися на різних частинах вхідних даних залежно від їх важливості). Також не малу роль грає сам датасет, який був значно розширений якраз з задумом, щоб модель BERT, використовуючи перевагу трансформерів, більш гарно зробила узагальнюючі висновки щодо всього спектру атак, які можуть бути в датасеті з розширеними характеристиками (тобто більш глибокий аналіз до розуміння). В цьому криється основна відмінність. І хоча робота [7] також використовує механізм уваги, але все одно використовується інша модель (не BERT), а також менший датасет, і в результаті може передбачуватись зовсім малий набір атак та зі зменшеною точністю (про це – в розділі з описом схожих робіт).

Насамперед, цей метод аналізує HTTP трафік, який можна представити у вигляді тексту (звичайний текст запиту), а далі даний текст буде класифікуватися моделлю, яка натренована на великій кількості прикладів як поганого (є атака), так і легітимного трафіку. Це дає змогу автоматично виявляти патерни в даних та будувати залежності між даними. А з використанням моделі BERT можна отримати перевагу в контекстному розумінні тексту. Тобто, залежності між даними будуть гарно знаходитися завдяки механізму уваги від трансформерів. Це основна перевага використання трансформерів, яка була описана у минулому підрозділі.

Далі буде наведено відмінності між реалізацією традиційних методів та реалізованого у ході дослідження. У традиційних методах дані зазвичай збираються з журналів серверів, мережевих пакетів та інших джерел. Вони часто представляються у вигляді рядків тексту або простих векторів ознак. В даному ж підході робиться все те саме, але відбувається й додаткова підготовка у вигляді токенізації, лематизації та видалення стоп-слів для забезпечення максимально точного представлення текстових даних.

Щодо векторизації слів, то в традиційних методах зазвичай це відбувається через так званий «мішок слів» (bag of words) або TF-IDF (term frequency-inverse document frequency), але такі методи не враховують контекстуальних зв’язків між словами, що може призводити до втрати важливої інформації. В запропонованому методі відбувається токенізації через трансформери, зокрема моделі BERT. Це дозволяє враховувати контекстуальні зв’язки між словами в запитах, що значно покращує точність представлення даних.

Навчання моделі в традиційних методах – зазвичай використовуються моделі типу логістичної регресії, випадкові ліси. Ці моделі добре працюють на простих задачах, але їх ефек-

тивність може знижуватися на складних даних через обмежені можливості врахування контексту. В цьому дослідженні використовується модель BERT, яка була донавчена на специфічних даних HTTP-запитів. Використання трансформерів дозволяє моделі краще розуміти контекст і взаємозв'язки між словами, що покращує її здатність ідентифікувати зловмисні запити.

Загалом, наведений підхід використовує контекстуальне розуміння тексту (тобто він розуміє контекст, що у звичайній розмові грає велику роль). Це дозволяє більш точно ідентифікувати аномалії та зловмисні дії. Завдяки цьому досягається високий рівень точності моделі, що було продемонстровано в ході експериментів. Про це – в наступному підрозділі.

Отже, в цьому розділі описано основні відмінності щодо запропонованого підходу з цієї роботи, а також підходів, які використовувалися в інших роботах. А також розглянуто перевагу цієї роботи над іншими, які могли так чи інакше також використовувати схожі архітектури, але різні моделі. Описано важливість розширення датасету.

Опис існуючих датасетів. Процес розширення датасету

Потрібно було зібрати та підготовити набір даних, що включав HTTP-запити як нормальні, так і зловмисні. Для цього спочатку було проаналізовано наявні датасети, а також можливість їх застосування.

Датасет від CSIC2010 [2] включає тисячі автоматично згенерованих веб-запитів та призначений для тестування систем захисту від веб-атак. Наявні в датасеті дані включають 36,000 нормальних та понад 25,000 аномальних запитів. Але в цьому датасеті була проблема, що він мав багато однакових даних, які відрізнялися тільки параметром сесії, що, як показала практика, ніяк не допомагає в класифікації трафіку (бо цей параметр сесії – це рандомне число і не більше). Тому з цього датасету було взято всі дані після видалення повторів.

Другий датасет, наданий PositiveTechnologies через GitHub [8], містить дані з 21,991 безпечних та 1,097 аномальних HTTP-запитів із банківської програми. Цей датасет є цінним ресурсом для вивчення та виявлення потенційних загроз у банківському середовищі. Він мав також свої дублюючі дані. Ще в нього була особливість, що перший рядок починався з дати та часу – відмітки часу, коли він був зібраний. Ці дані абсолютно були непотрібні, тому датасет треба було обробити та прибрати непотрібне.

Загалом, на основі цих двох датасетів спочатку було сформовано першу версію датасету з 58 тисяч даних. Було поділено на тренувальну (40 тис.), тестувальну (13 тис.) та валідаційну (5 тис.) вибірки. Це робиться для того, щоб тренувати модель на тренувальній, а також паралельно перевіряти на тестувальній. І вже після тренування провести фінальне випробування на валідаційній вибірці, яку модель «не бачила» під час тренування, а отже не могла брати з неї ніякі залежності, не вивчала ці дані і тд. Це дає більш об’єктивний спосіб перевірити результати.

Під час тренування стало очевидно, що сформованого датасету недостатньо для створення гарної узагальнюючої моделі (в порівнянні моделей це описано), тож, було взято третій датасет від openappsec [9], який є доволі великим та включає в себе 973,964 легітимних HTTP-запитів з 185 реальних веб-сайтів у 12 категоріях, а також 73,924 зловмисних завантажень. В ньому також видалені дублікати (майже кожен п’ятий запис), але їх було видалено і сформовано другу версію датасету зі 195 тис. записів. Де тренувальна включала 117 тис., тестувальна – 51 тис., і валідаційна – 27 тис. записів.

Також в ході формування датасету було використано багато різних маленьких джерел в інтернеті, з яких бралася певна кількість даних для доповнення фінального датасету.

Отже, далі всі датасети було скомбіновано, видалено повторення, а також певні дані, які містили помилки. В результаті вийшло, що всі датасети збалансовані, мають середнє відхилення 0,5 та середнє значення 0,5 за колонкою «benign», яка містить значення «1» або «0» в залежності від того, є аномалія, чи її немає. Наступним етапом треба було тренувати моделі, процес чого описано в наступному розділі.

Хід проведення експериментів

Експерименти проводилися з метою розробки та оцінки ефективності класифікатора для виявлення веб-атак через аналіз HTTP-трафіку з використанням методів обробки природної мови (NLP) та моделей на базі трансформерів, зокрема BERT. Для цього використовувався датасет, методику збору якого описано у попередньому розділі.

Для підготовки даних використовувалися такі методи: токенізація, лематизація та видалення стоп-слів. Після цього HTTP-запити були представлені у вигляді векторів, придатних для обробки моделлю BERT. Модель BERT була обрана за її здатність до контекстного розуміння тексту, що є критичним для ідентифікації складних шаблонів у HTTP-запитах.

Експерименти проводилися в кілька етапів. На першому етапі модель BERT тренувалася на підготовленому наборі даних з метою класифікації запитів як нормальних або зловмисних. Для тренування використовувалися стандартні методи оптимізації, такі як градієнтний спуск, та функції втрат, що підходять для задач бінарної класифікації.

Результати тренування чотирьох моделей на підготовлених датасетах можна побачити на рис. 2 (графіки двох моделей «bert-tiny-4.39m-sql-e70» та «bert-tiny-4.39m-sql-e50» майже повністю співпадають). На графіку відображено як змінювалась похибка моделі (loss, вертикальна вісь) в залежності від кроків навчання (train step), які змінюються по горизонтальній осі. Як видно, модель «bert-tiny-4.39m-nosql-e1-extended» стала найкращою серед усіх (має найменшу похибку). Вона навчалася на розширеному датасеті, який включав більше даних, ніж попередні.

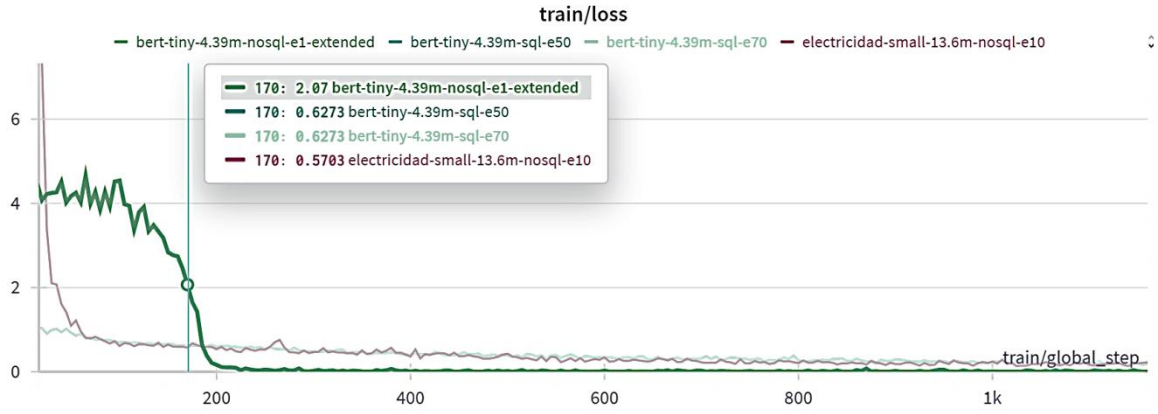

Рис. 2. Результати тренування моделей

На рис. 2 відображено графік, який автоматично сформовано під час навчання з використанням бібліотеки wandb мови Python. Ця бібліотека дозволяє відслідковувати параметри під час навчання, дивитися проміжні та фінальні результати. Для повторення експерименту необхідно використовувати саме цю бібліотеку, яка допоможе автоматично сформувати даний графік після навчання моделі. По вертикальній осі відображена похибка (це просто певне число) у вигляді функції втрат (різниця між кінцевим результатом передбачення моделі та справжнім результатом. Це просто число, і в ході навчання воно повинне зменшуватися, що і видно на рисунку), а горизонтально наведено шаг (або ітерацію) тренування моделі. Різні початкові точки графіків для різних моделей можна пояснити різними датасетами, що використовувались при навчанні, а також різними внутрішніми параметрами для різних моделей.

Вертикальна блакитна лінія – це просто графічний елемент, який було отримано наведенням курсора миші на довільну ділянку графіку. Вона допомагає відображати певний момент на графіку, щоб можна було побачити результати моделі в цій точці, а також виділити «товстим» лінію з найкращою моделлю у відповідному редакторі для наочності.

Самі по собі моделі відрізняються лише даними, на яких вони тренувалися. Це потрібно, щоб знайти баланс між необхідною кількістю даних для тренування, а також отриманою

помилкою. Назви моделей «bert-tiny-4.39m-sql-e70» та «bert-tiny-4.39m-nosql-e50», а також модель «electricidad-small-13.6m-nosql-e10» мають певний сенс в плані найменування, бо відображають дані, на яких тренувалися, та інші параметри. В їх назві є кількість параметрів (4,39 та 13,6 мільйонів), кількість епох «e70», а також примітка «sql» або «nosql». Ця примітка показує, чи були в тренувальному датасеті моделі дані нормального трафіку, який містив нормальний SQL запит до бази даних. Фінальний датасет для тренування налічував 195 тис. запитів та був збалансований (тобто, там була майже однакова кількість прикладів поганого та хорошого трафіку). Цей датасет розбивався на 117 тис. тренувальних даних, а також на 51 тис. тестувальних і 27 тис. – валідаційних. На тренувальних відбувалося тренування і використовувалися тестувальні дані для корекції під час тренування, а далі після тренування на готовій моделі валідаційний датасет використовувався як фінальна перевірка результату моделі (але це не той датасет зі статті [9], на якій перевірялася остаточна точність, бо це лише датасет для перевірки проміжних результатів для відбору кращих серед побудованих моделей). Тренування відбувалося завдяки можливостям мови програмування Python. Воно брало певну порцію даних, навчалося на ній, а далі брався тестувальний датасет і проганявся по моделі, видаючи похибку, яку можна бачити на рис. 2. Таким чином відбувалося до тих пір, поки похибка не буде зведена до мінімуму.

Також з рис. 2 можна бачити, що фінальна модель «bert-tiny-4.39m-nosql-e1-extended» (навчалася на розширеному датасеті) на початку свого тренування мала більшу похибку, аніж інші моделі, такі як «bert-tiny-4.39m-sql-e70» та «bert-tiny-4.39m-sql-e50» (навчалися на вузькому датасеті). Цей момент показує важливість навчання на більшому датасеті, бо моделі, які навчалися на зменшеному дата сеті, мають меншу узагальнену властивість. Тому що, коли датасет розширили та почали навчати іншу модель (відбувалося «донавчання»), то така модель почала показувати багато помилок (бо взяла менше характеристик з малого датасету).

Отже, в цьому розділі описано процес, в ході якого проводилися основні експерименти, а також результати щодо побудованих моделей. Обрано кращу модель. Наведено відомості щодо важливості використання збільшеного датасету і як це відображається на загальній картині тренування та якості моделі. В наступному розділі наведено процес оцінки та порівняння моделей між собою, що передбачає визначення основних метрик для оцінки, а також порівняння на базі цих метрик побудованих моделей з іншими фаєрволами, які використовуються в промисловості. Наведено числові результати ефективності побудованого класифікатору для аналізу веб-трафіку та проаналізовано ці результати з результатами інших методів.

Процес оцінки та порівняння моделей між собою

Після тренування модель тестували на незалежному наборі даних, щоб оцінити її ефективність. Далі необхідно порівняти побудовану модель з класичними методами, які вже зарекомендували себе та використовуються на ринку, таким чином підтвердити або спростувати поставлену на початку роботи гіпотезу. Тому для порівняння ефективності побудованої моделі було також використано результати тестування з роботи [9], де порівняно між собою дев'ять провідних рішень систем захисту веб-додатків (WAF). Результати кожної системи порівнювалися з результатами запропонованої моделі, щоб визначити її переваги та недоліки. На завершальному етапі експериментів проведено аналіз помилок, щоб ідентифікувати випадки, коли модель помилялася, та визначити можливі напрямки для подальшого вдосконалення. Цей етап включав детальний розгляд неправильно класифікованих запитів та аналіз їхніх особливостей. Таким чином, проведені експерименти дозволили не тільки підтвердити гіпотезу про перевагу машинного навчання над традиційними методами, але й надати конкретні рекомендації щодо подальшого розвитку методів виявлення веб-атак.

Ключовою метрикою є збалансована точність. Вона отримується як середнє між метриками TNR (true negative rate), або її ще називають Specificity, та метрикою TPR (true positive rate), або Recall.

Метрика TNR, також відома як Specificity, є однією з метрик, яка використовується для оцінювання ефективності класифікаційних моделей, особливо в задачах двокласової класифікації. Вона вимірює здатність моделі правильно ідентифікувати всі дійсні негативні приклади з усіх доступних негативних прикладів у наборі даних. Іншими словами, TNR показує, яку частку реальних негативів (об'єктів, що не належать до класу, який нас цікавить) модель змогла правильно передбачити.

Метрика TPR (або recall) вимірює здатність моделі правильно ідентифікувати всі дійсні позитивні приклади з усіх доступних позитивних прикладів у наборі даних. Простіше кажучи, recall показує, яку частку реальних позитивів (об'єктів, що належать до класу, який нас цікавить) модель змогла коректно передбачити. Формально, recall визначається як відношення кількості правильно передбачених позитивних випадків (True Positives, TP) до суми дійсних позитивних випадків, що складається з правильно передбачених позитивів (TP) та неправильно передбачених негативів (False Negatives, FN).

Загалом, їх потрібно отримувати, щоб вирахувати збалансовану точність, яка всього лише є середнім значенням між Specificity та Recall. Ця метрика дозволяє комплексно оцінити здатність моделі правильно класифікувати зловмисні запити, мінімізуючи хибні спрацювання.

Для порівняння з існуючими фаєрволами було обрано кращу навчену модель та порівняно за основними параметрами з метою зрозуміти, чи буде така модель краще працювати, чи ні. Для цієї мети підходить стаття [9], яка досліджує якість сучасних фаєрволів на конкретному датасеті, який було використано і в цій роботі. У рамках дослідження автори вирішили провести глибокий, але простий експеримент, спрямований на виклик як шкідливих, так і легітимних веб-запитів у різних веб-захистах та вимірювання їх результатів. Для проведення тесту використовувався дуже великий набір даних: 973,964 легітимних HTTP-запитів з 185 реальних веб-сайтів у 12 категоріях, а також 73,924 зловмисних запитів з широкого спектру загроз, які зазвичай зустрічаються в атаках.

Експеримент, проведений авторами статті [9] у липні 2023 р., порівнював кілька популярних рішень WAF, серед яких були Microsoft Azure WAFv2 з правилами OWASP CRS 3.2, AWS WAF з керованим набором правил AWS, AWS WAF з керованим набором правил AWS та F5 Ruleset, CloudFlare WAF з керуванням та OWASP Core Rulesets, F5 NGINX App Protect WAF з профілем за замовчуванням, F5 NGINX App Protect WAF зі строгим профілем, NGINX ModSecurity з правилами OWASP CRS 3.3.4, open-appsec / CloudGuard AppSec з конфігурацією за замовчуванням (High Confidence), а також open-appsec / CloudGuard AppSec з критичною конфігурацією впевненості.

Результати тесту підкреслюють значущі відмінності у продуктивності різних захисних рішень. Наприклад, CloudFlare WAF продемонстрував майже ідеальну якість виявлення (99,945 %), але найнижчу якість безпеки (67,297 %) серед усіх перевірених продуктів. Azure WAF забезпечує дуже високу якість безпеки (98,547 %), але також має надзвичайно високий рівень хибнопозитивних результатів (38,346 %). Такі результати вказують на значний ризик безпеки або необхідність ретельного налаштування як спочатку, так і в подальшому для ефективного використання продукту в реальних умовах.

Для врахування дисбалансу в кількості екземплярів у різних класах у цьому порівнянні також використали метрику Balanced Accuracy, яка враховує середню точність по кожному класу. Результати порівняння по збалансованій точності (ЗТ) можна побачити на наступному рисунку. Автори зазначають, що open-appsec / CloudGuard AppSec зі стандартною конфігурацією забезпечує найвищий показник збалансованої точності (97,28 %), за ним іде цей же продукт із конфігурацією Critical Profile (ЗТ – 96,8 %), а також NGINX AppProtect зі строгим профілем (ЗТ – 92,52 %). На рис. 3 [9] можна побачити кількісне значення збалансованої точності, яке було отримано для того чи іншого фаерволу в ході оцінки його точності на перевірочному датасеті.

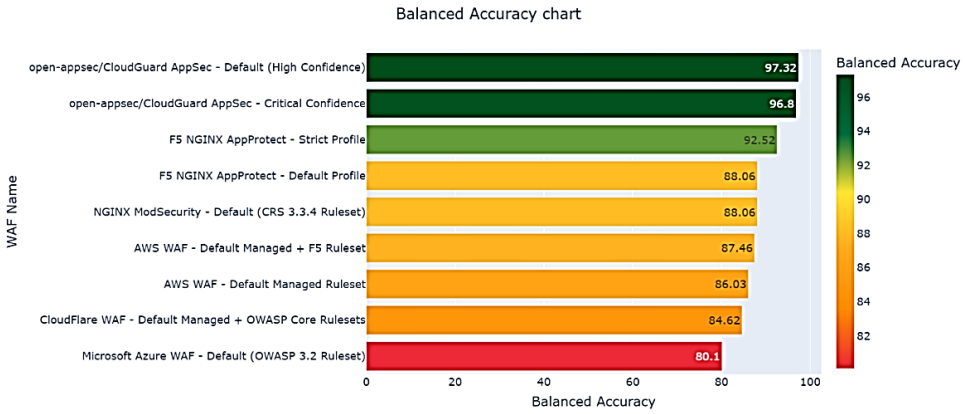

Рис. 3. Наявні метрики звичайних фаєрволів для порівння [9]

Експеримент в рамках цього дослідження проводився на тому ж самому датасеті, що й від авторів статті [9] (використовувався вкрай обширний набір даних: 973,964 легітимних HTTP-запитів з 185 реальних веб-сайтів у 12 категоріях, а також 73,924 зловмисних запитів з широкого спектру загроз, які зазвичай зустрічаються в атаках). А саме було взято найкращу натреновану модель під назвою «bert-tiny-4.39m-nosql-e1-extended» та зроблено спроби передбачити по всьому датасету чи то нормальний трафік, чи ні.

В результаті отримано метрики, з яких можна отримати збалансовану точність (основная метрика для порівняння). Створена модель демонструє Recall (Sensitivity) на рівні 0,9997 та Specificity на рівні 1,0. Тоді середнє значення цих показників, яке є збалансованою точністю, дорівнює 0,9998. Це значення значно перевищує найкраще рішення з порівняння зі статті [9], яке, за словами авторів, має збалансовану точність на рівні 0,9732. Тобто, наведений метод з використанням моделі транформерів BERT можна цілком назвати конкурентоспроможним завдяки його здатності розуміти контекст та автоматично вивчати зв'язки в даних під час тренування. До недоліків можна віднести необхідність мати датасет для навчання моделі, а також обчислювальні потужності, бо тренування займає певний час і може виникнуть необхідність мати потужну GPU для цієї задачі.

Висновки

Проведене дослідження демонструє значні можливості використання методів машинного навчання та обробки природної мови (NLP) для підвищення ефективності виявлення веб-атак. Використання моделі BERT, заснованої на трансформерах, дозволило створити високоефективний класифікатор для аналізу HTTP-трафіку.

Результати експериментів показали, що запропонована модель здатна з більшою точністю ідентифікувати зловмисні запити порівняно з традиційними методами, що базуються на правилах і сигнатурах.

Результати дослідження можуть бути використані для вдосконалення існуючих систем кібербезпеки, сприяючи розробці більш надійних і ефективних рішень для захисту від веб-атак. Найкраща модель, натренована в рамках цього дослідження під назвою bert-tiny-4.39m-nosql-e1-extended, демонструє чутливість (Recall) на рівні 0,9997 та специфічність (Specificity) на рівні 1,0. Середнє значення цих показників, яке є збалансованою точністю, становить 0,9998. Це значення значно перевищує найкраще рішення, описане у статті [9], де автори зазначають збалансовану точність на рівні 0,9732.

Порівняння з іншими роботами показало, що наша модель має кілька ключових переваг.

По-перше, модель BERT використовує сучасний підхід до токенізації, зокрема WordPiece токенізатор, який дозволяє краще обробляти рідкісні та незнайомі слова. На відміну від токенізаторів у інших роботах, WordPiece забезпечує розбиття слів на підслова, що підвищує ефективність обробки тексту та узагальнюючу здатність моделі.

По-друге, BERT забезпечує глибоке контекстне розуміння завдяки своєму глобальному механізму уваги, що дозволяє моделі враховувати взаємозв'язки між словами на всіх рівнях тексту. Це є значною перевагою над методами, які використовують Doc2Vec, LSTM-CNN, Isolation Forest або ансамблі з методами NLP, що не враховують глибокий контекст або обмежуються локальними зв'язками в тексті.

По-третє, наша модель була навчена на великому та збалансованому наборі даних, який включає кілька датасетів, таких як CSIC2010, PositiveTechnologies, openappsec та власноруч зібрані дані. Це значно покращило узагальнюючу здатність моделі та її точність у виявленні складних веб-атак.

Порівняння з дев'ятьма провідними системами захисту веб-додатків (WAF) підтвердило конкурентоспроможність та переваги запропонованого методу. Гіпотеза підтвердилася.

Отже, створена модель демонструє високу збалансовану точність та здатність розпізнавати складні веб-атаки з мінімальною кількістю хибних спрацювань, що робить її конкурентоспроможною серед існуючих рішень у цій сфері. Враховуючи ці переваги, модель BERT може бути рекомендована для використання у системах виявлення веб-атак для забезпечення високого рівня безпеки та надійності.

Дослідження робить важливий внесок у розвиток методів виявлення веб-атак, демонструючи, що інтеграція передових технологій NLP та моделей машинного навчання може суттєво підвищити рівень кібербезпеки в умовах постійно зростаючих загроз.

Список літератури:

1. Attention Is All You Need [Електронний ресурс]. Режим доступу до ресурсу: https://arxiv.org/pdf/1706.03762.pdf
2. HTTP DATASET CSIC 2010 [Електронний ресурс]. Режим доступу до ресурсу: https://www.isi.csic.es/dataset/
3. Saikat Das, Mohammad Ashrafuzzaman, Frederick T Sheldon, and Sajjan Shiva. Network intrusion detection using natural language processing and ensemble machine learning // 2020 IEEE Symposium Series on Computational Intelligence (SSCI), pages 829–835. IEEE, 2020.
4. Ali Moradi Vartouni, Saeed Sedighian Kashi, and Mohammad Teshnehlab. An anomaly detection method to detect web attacks using stacked auto-encoder // 2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS), pages 131–134. IEEE, 2018.
5. Jiaxin Liu, Xucheng Song, Yingjie Zhou, Xi Peng, Yanru Zhang, Pei Liu, and Dapeng Wu. Deep anomaly detection in packet payload. arXiv preprint arXiv:1912.02549, 2019.
6. Chaochao Luo, Zhiyuan Tan, Geyong Min, Jie Gan, Wei Shi, and Zhihong Tian. A novel web attack detection system for Internet of things via ensemble classification // IEEE on Industrial Informatics, 2020
7. Tianlong Liu, Yu Qi 2, Liang Shi, Jianan Yan. Locate-Then-Detect: Real-time Web Attack Detection via Attention-based Deep Neural Networks
8. mrm8488/bert-tiny-finetuned-sms-spam-detection [Електронний ресурс]. Режим доступу до ресурсу: https://huggingface.co/mrm8488/bert-tiny-finetuned-sms-spam-detection
9. Best WAF solutions in 2023 - real-world comparison [Електронний ресурс]. Режим доступу до ресурсу: https://www.openappsec.io/post/best-waf-solutions-in-2023-real-world-comparison

Надійшла до редколегії 05.09.2024

Відомості про авторів:

Кавецький Максим Сергійович – Харківський національний університет радіоелектроніки, магістр кафедри безпеки інформаційних технологій факультет комп'ютерної інженерії та управління; Україна; e-mail: maksym.kavetskyi@nure.ua; ORCID: https://orcid.org/0009-0008-7419-1029

Руженцев Віктор Ігорович – д-р техн. наук, доцент, Харківський національний університет радіоелектроніки, професор кафедри безпеки інформаційних технологій; Україна; e-mail: viktor.ruzhentsev@nure.ua, ORCID: http://orcid.org/0000-0002-1007-6530