МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри Комп'ютеризованих
систем захисту інформації

_____ Михайло СТЕПАНОВ
«____»_____ 2023 р.

На правах рукопису
УДК 004.056.5:510.22(043.3)

КВАЛІФІКАЦІЙНА РОБОТА

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»

Тема: «Інтелектуальні моделі класифікації подій кібербезпеки»

Виконавець: Олександр САВЧУК
Керівник: д.т.н., с.н.с. Михайло СТЕПАНОВ

Консультант розділу «Охорона навколишнього середовища»: к.т.н., доцент Тетяна ДМИТРУХА

Нормоконтролер: к.т.н., доцент Анна ІЛЬЄНКО

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: Кібербезпеки та програмної інженерії

Кафедра: Комп'ютеризованих систем захисту інформації

Освітній ступінь: Магістр

Спеціальність: 125 «Кібербезпека»

Освітньо-професійна програма: «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри Комп'ютеризованих систем захисту інформації

_____ Михайло СТЕПАНОВ

«__» _____ 2023 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

здобувача вищої освіти Савчука Олександра Васильовича

1. Тема: «*Інтелектуальні моделі класифікації подій кібербезпеки*» затверджена наказом ректора від «15» вересня 2023 р. № 1814/ст.
2. Термін виконання: з 16.10.2023 р. по 31.12.2023 р.
3. Вихідні дані: завдання та план на дослідження, перелік основної літератури.
4. Зміст пояснювальної записки: аналіз літературних джерел, вибір нейронної мережі, моделювання роботи, аналіз результатів моделювання.

5. КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

№ з/п	Етапи виконання кваліфікаційної роботи	Термін виконання етапів	Примітка
1.	Уточнення постановки задачі	16.10.2023	*Виконано*
2.	Аналіз літературних джерел	20.10.2023	*Виконано*
3.	Обґрунтування вибору рішення	21.10.2023	*Виконано*
4.	Збір інформації	25.10.2023	*Виконано*
5.	Вибір нейронної мережі	30.10.2023	*Виконано*
6.	Розробка програмного забезпечення	04.11.2023	*Виконано*
7.	Моделювання нейронної мережі	20.11.2023	*Виконано*
8.	Створення пояснювальної записки	5.12.2023	*Виконано*
9.	Перевірка на антиплагіат	10.12.2023	*Виконано*
10.	Оформлення і друк пояснювальної записки	15.11.2023	*Виконано*
11.	Оформлення презентації	20.12.2023	*Виконано*
12.	Отримання рецензій від рецензента	22.12.2023	*Виконано*

6. Консультанти з окремих розділів

Розділ	Консультант (посада, П.І.Б.)	Дата, підпис	
		Завдання видав	Завдання прийняв
Охорона навколишнього середовища	Дмитруха Т.І.		

7. Дата видачі завдання: «16» жовтня 2023 р.

Здобувач вищої освіти Олександр САВЧУК

(підпис, дата)

Керівник кваліфікаційної роботи Михайло СТЕПАНОВ

(підпис, дата)

РЕФЕРАТ

Випускна робота «Інтелектуальні моделі класифікації подій кібербезпеки» складається із вступу, основної частини, що містить 3 розділи, висновків, списку використаних джерел та додатків. Загальний обсяг роботи – 99 сторінок. Робота містить 16 малюнків, 18 таблиць та 2 додатки.

Об'єкт дослідження - процес класифікації подій кібербезпеки із застосуванням інтелектуальних моделей.

Мета роботи - розробка моделі захисту від кібератак із використанням нейронних мереж.

Предмет дослідження - моделі класифікації подій кібербезпеки на основі нейронних мереж.

Метод дослідження - синтез та аналіз моделей класифікації подій у мережі кібербезпеки з використанням методу імітаційного моделювання.

Практична цінність одержаних результатів складається з розробки інтелектуальної моделі для ідентифікації та класифікації мережевих атак, які можна використовувати як компоненти системних прийняття рішень щодо захисту бізнесу, таких як WAF (міжмережевий екран веб-додатків). Використання розроблених моделей дозволяє підвищити надійність виявлення загроз у поєднанні з наявними засобами захисту.

Наукова новизна дослідження. В роботі створено модель виявлення загроз інформаційній безпеці, які відрізняються від сучасних аналогів формою аналізу даних, швидкістю дії та поданням результатів.

Ключові слова: нейронні мережі, моделі інтелектуального аналізу даних, ідентифікація атак, виявлення вторгнень, DoS-атаки, машинне навчання.

ЗМІСТ

ВСТУП

Глобалізація інформації, яка зумовлена поширенням Інтернету, є однією з проявів глобальної інтеграції світового простору. Зростання обсягів атак і зловживань у сфері інформаційних технологій є невід'ємною частиною цього процесу, що призводить до збільшення уваги до питань безпеки комп'ютерних систем.

Більшість традиційних методів захисту від кібератак не забезпечують достатнього рівня захисту через їхню неефективність у боротьбі з новими атаками, низьку точність та обмежену швидкість реакції. Таким чином, на сучасному етапі активно розробляються різноманітні технології захисту комп'ютерних мереж, засновані на інтелектуальному аналізі даних, використанні нейронних мереж, статистичному аналізі та інших підходах.

Ефективна стратегія запобігання кіберзагрозам (Cyber Threat Intelligence – CTI) передбачає виявлення майбутніх загроз, поведінки та намірів зловмисників. Проактивний характер стратегії спрямований на прогнозування та усунення вразливостей та експлойтів, що значно підвищує ефективність реагування на атаки у реальному часі. CTI представляє собою складну область, яка застосовується в різних контекстах.

У останні п'ять років використання штучного інтелекту та машинного навчання для збору даних для розвідки стрімко зросло. ABI Research передбачає, що "витрати на машинне навчання в галузі кібербезпеки зростуть до 96 мільярдів доларів до 2021 року". З урахуванням різноманітних джерел даних в кіберінфраструктурі, машинне навчання може ефективно використовуватися для виявлення системних аномалій, ботнетів та фішингу. Збір інформації про загрози є першою лінією оборони в інфраструктурі кібербезпеки, а реактивні системи, такі як системи виявлення вторгнень (IDS), входять у другу лінію захисту.

Об'єкт дослідження - процес класифікації подій кібербезпеки із застосуванням інтелектуальних моделей.

Мета роботи - розробка моделі захисту від кібератак із використанням нейронних мереж.

Предмет дослідження - моделі класифікації подій кібербезпеки на основі нейронних мереж.

Практична цінність отриманих результатів полягає в розробці інтелектуальних моделей для ідентифікації та класифікації мережевих подій, які можуть бути використані як компоненти системних рішень забезпечення безпеки бізнесу, таких як WAF (міжмережевий екран веб-додатків) та система IEM. Використання розроблених моделей дозволяє підвищити надійність виявлення загроз у поєднанні з наявними засобами захисту.

Наукова новизна дослідження полягає в створенні нетипових моделей виявлення загроз інформаційній безпеці, які відрізняються від сучасних аналогів формою аналізу даних, швидкості її дії.

РОЗДІЛ 1 АНАЛІЗ ТА ПОСТАНОВА ЗАДАЧИ

1.1 Аналіз випадків кіберзлочинності та заходи протидії

Кількість кіберзлочинів, пов'язаних із використанням інформаційних систем для вирішення бізнес-процесів, зростає щорічно. Це явище пояснюється збільшенням обсягу засобів автоматизації та ростом інтересу певних груп населення до отримання та контролю корисної інформації.

Цільові атаки на державні ресурси, правоохоронні системи, банківські установи та об'єкти критичної інфраструктури поширюються як в Україні, так і в інших країнах. Крім хакерських груп, які вчиняють масові кібератаки, в кіберпросторі активно розвиваються традиційні організовані злочинні угруповання. Це пов'язано з недостатньою ефективністю систем захисту від вторгнень, високою ймовірністю анонімності в Інтернеті та доступністю розподілених апаратних ресурсів за невисокими цінами.

Кожна кіберугруповання має свою мотивацію, методи виконання атак і може переслідувати не лише особисті цілі, але й прагнення до справедливості. Прикладом такої атаки є #OpISIS – кібератака на терористичну мережу ІДІЛ після терактів у Парижі у 2015 році.

Ще однією перспективою вивчення є фокус на групах як об'єктах злочинів. Дослідження поведінки молоді в Інтернеті є поширеною концепцією, оскільки вони є особливо вразливою групою. Релігійна, расова та етнічна дискримінація також породжують численні злочини в Інтернеті, включаючи утиск та розпалювання ненависті.

З'явлення Інтернет-груп і спільнот, які взаємодіють у віртуальному середовищі, відбулося з моменту створення Інтернету. Перші групи з'явилися на платформах, таких як Internet Relay Chat (IRC) та чатах AOL. Групи розповсюдилися на соціальні мережі, месенджери та інші інтернет-сервіси. Зростання кількості користувачів

соцмереж і месенджерів створило умови для формування кібер- та кіберзлочинних груп в Інтернеті.

Форуми кіберзлочинців та терористів часто знаходяться в "темній мережі", яка існує в зашифрованій мережі, до якої можна отримати доступ за допомогою спеціального програмного забезпечення, наприклад, Tor та I2P. Це забезпечує певний рівень анонімності, що привертає злочинців. Форуми даркнета та їхні комунікації є об'єктом уваги дослідників кіберзлочинності для передбачення та підготовки до захисту від можливих кібератак.

Організовані кібератаки стосуються не тільки державних установ, але й підприємств різного розміру. Невизначеність можливості стати об'єктом цільової атаки примушує підприємства витрачати гроші на засоби безпеки. Нерозуміння ризиків може призвести до великих втрат, включаючи втрату даних, фінансові втрати та втрату репутації. Малі компанії, які стали об'єктом кібератак, часто змушені припинити свою діяльність. Наприклад, 71% зломів баз даних відбувається в малих підприємствах.

Прогнозується, що збитки від кіберзлочинності складуть 6 мільярдів доларів щорічно до 2021 року. Важливо розглядати головні світові тенденції в цьому сегменті для правильного реагування на загрози.

Згідно з результатами «Звіту про розслідування порушень даних Verizon», що ґрунтується на реальних даних про 42 789 інцидентів інформаційної безпеки та 2015 витоків даних, наданих 83 джерелами, як державними, так і приватними, охоплюючи 86 країн світу, можна виокремити наступну динаміку поширення у різних галузях людської діяльності за 2020 рік, що відображена у таблицях 1.1 – 1.9 [1]. Важливо відзначити, що найбільше інцидентів зафіксовано в галузі державного управління, тоді як сфера послуг готелів та ресторанів виявилася менш схильною до кібератак, при чому загальна кількість інцидентів зменшилася в порівнянні з минулими роками. Зазначимо, що більшість кібератак в цій галузі мали фінансовий мотив, тоді як шпигунські та розважальні мотивації були менш поширеними.

Таблиця 1.1 — Статистика інцендентів

Кількість інцидентів знизилася, порівняно з попередніми періодами. Це пов'язано з відсутністю інцидентів з постачальниками POS-терміналів, які раніше призводили до компрометації даних партнерів, що їх використовують.	
Частота	97 інцидентів, 68 підтверджений витік даних
Базові сценарії	На фізичні вторгнення в точки продажу, зламування веб-додатків і поширення шкідливого ПЗ припадає 95% всіх виявлених інцидентів.
Модель злочинця	Зовнішній (96%), внутрішній (7%)
Мотивація	Фінансовий (95%-100%)
Компрометація даних	Дані платіжної картки (87%), дані рахунку (13%), внутрішня інформація організації (25%)

Таблиця 1.2- Фінансова статистиката страхові випадків

Відмова в обслуговуванні та використання вкрадених облікових записів у банківських додатках, як і раніше, залишаються поширеними сценаріями. Скомпрометовані облікові записи електронної пошти виявляються лише після встановлення особи зловмисника. Шахрайство зі скімінгом у банкоматах продовжує скорочуватися.	
Частота	937 інцидентів, 226 підтверджених витоків даних
Базові сценарії	Злами веб-додатків, підвищення привілеїв та різні програмні помилки становлять 73% усіх виявлених інцидентів.
Модель злочинця	Зовнішні (82%), внутрішні (35%), обидві сторони (13%), партнери (5%)
Мотивація	Фінансовий (85%), шпигунство (15%)
Компрометація даних	Персональні дані (45%), облікові дані (48%), внутрішня інформація (48%)

Таблиця 1.3- Статистика освітипослуги

Освітні послуги, як і раніше, страждають від помилок у програмному забезпеченні, соціальній інженерії та недостатній безпеці облікових даних електронної пошти. За кількістю інцидентів DoS-атаки становлять понад половину всіх інцидентів.	
Частота	393 інциденти, 96 підтверджених витоків даних
Базові сценарії	Злами веб-додатків та помилки різного типу становлять 83% усіх виявлених інцидентів.
Модель злочинця	Зовнішній (61%), внутрішній (48%), двосторонній (3%)
Мотивація	Фінансовий (81%), шпигунство (14%), для розваги (5%), для невдоволення (3%), ідеологічний (3%)
Компрометація даних	Персональні дані (56%), облікові дані (58%), внутрішня інформація (38%)

Таблиця 1.4– Статистика щодо системи охорони здоров'я

Системи охорони здоров'я виділяються і натомість інших, оскільки більшість порушень пов'язані з внутрішніми структурами. Атаки типу «відмова в обслуговуванні» трапляються рідко, але проблеми з доступністю часто виникають через програми-вимагачі.	
Частота	476 інцидентів, 315 підтверджених витоку даних
Базові сценарії	Злами веб-додатків, підвищення привілеїв та різні помилки становлять 83% усіх виявлених інцидентів.
Модель злочинця	Внутрішні (50%), зовнішні (43%), партнери (5%), групи зацікавлених сторін (2%)
Мотивація	Фінансові (85%), для розваги (5%), вигідного становища (3%), для невдоволення (4%), шпигунства (3%),
Компрометація даних	Медичні дані (61%), персональні дані (24%), дані облікового запису (15%)

Розуміння основних типів кібератак є важливим аспектом. Зображення класифікації за групами представлене на рис. 1.2. Зростання кількості обширних кібератак свідчить про тенденцію до збільшення як кількості, так і серйозності інцидентів у сфері кібербезпеки. Витоки інформації порушують конфіденційність та можуть призвести до компрометації особистих даних користувачів, що в свою чергу призводить до пошкодження репутації компанії та її позиції на ринку праці.

Таблиця 1.5– Статистика з інформаційних послуг

Більшість веб-програм зазнають атак, які мають на меті порушення доступності та отримання доступу до хмарних облікових записів та адрес електронної пошти організацій.	
Частота	1084 інциденти, 165 підтверджених витоків даних
Базові сценарії	Злами веб-додатків, помилки різного типу та кібершпигунство є причиною 85% всіх виявлених інцидентів.
Модель злочинця	Зовнішній (53%), внутрішній (44%), партнерський (2%)
Мотивація	Фінансові (66%), шпигунство (30%)
Компрометація даних	Персональні дані (47%), облікові дані (32%), конфіденційна інформація (24%)

Рисунок1.1- Кіберзлочинність у 2020 року

Важливо розуміти основні типи кібератак, що відбуваються. Розріз за групами представлений на рис. 1.2.

Збільшення кількості великомасштабних кібератак демонструє тенденцію до збільшення кількості інцидентів безпеки, як за кількістю, так і серйозністю. Витіки інформації порушують конфіденційність та призводять до компрометації персональних даних користувачів, що, у свою чергу, руйнує репутацію компанії та її становище на ринку праці.

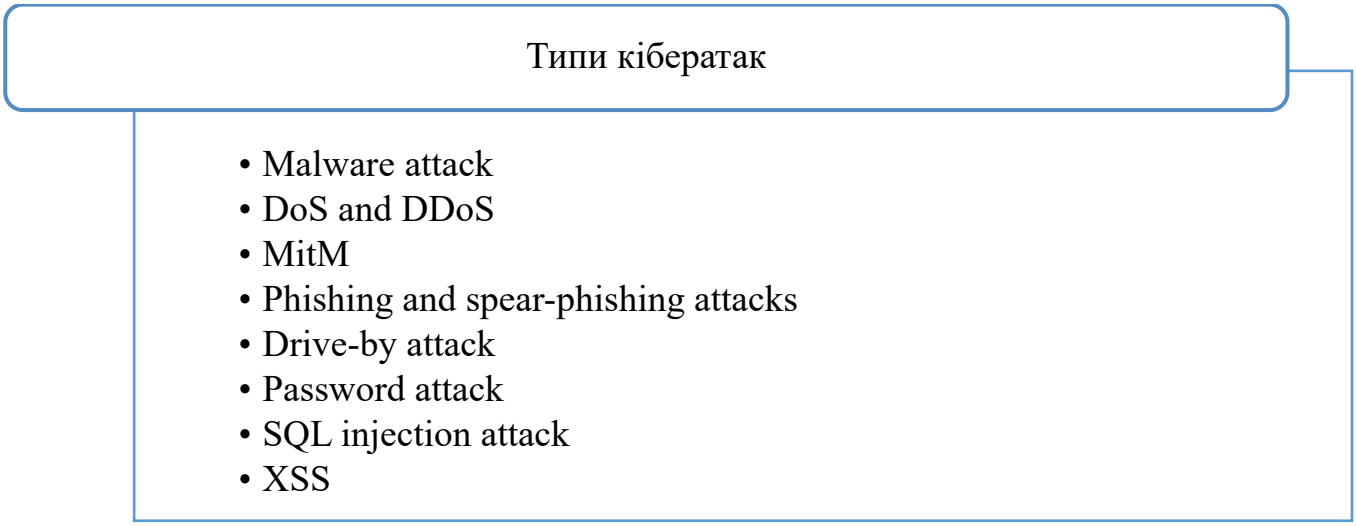

Рисунок 1.2- Основні види кібератак

Можна виокремити низку фактів, що висвітлюють тенденції та наслідки успішних кібератак:

- Кількість інцидентів у сфері кібербезпеки зросла на 12% від 2019 року та на 68% від 2015 року.
- Хакерські атаки відбуваються кожні 40 секунд, що загалом становить 2238 подій щодня [6].
- Середній термін виявлення інциденту витоку даних у 2020 році становив 6 місяців за даними IBM.
- Середній життєвий цикл інциденту триває майже 10 місяців, враховуючи час від початку до локалізації (IBM).
- Середня вартість витоку даних у 2020 році складала $4 млн [17].

- У 2017 році Uber повідомив про крадіжку інформації від понад 57 мільйонів клієнтів та водіїв, і зловмисники змусили компанію виплатити більше 100 000 доларів.
- Незважаючи на те, що витік даних з американського офісу Equifax стався у 2018 році, компанія досі оплачує $4,2 млрд.

Аналізуючи статистику кіберзлочинності, можна зробити такі висновки:

- 35% кіберінцидентів сталися за участю внутрішніх суб'єктів.
- 70% організацій не вірять у можливість блокування потенційних загроз антивірусним програмним забезпеченням.
- 90% шкідливого програмного забезпечення поширюється електронною поштою.
- 41% жертв інцидентів припадають на малі та середні компанії.
- 16% інцидентів вплинули на сферу охорони здоров'я, 11% – на фінансовий сектор, та 17% – на державний сектор.
- Банківський сектор зазнав найбільших витрат, пов'язаних з кіберзлочинністю, у 2019 році — $19 млн.
- Імовірні збитки сектору охорони здоров'я у 2020 році склали 27 мільярдів доларів США.

Широка поширеність атак на інформаційні системи у глобальному кіберпросторі вимагає підготовки та постійного оновлення систем захисту. Інформація щодо витрат на кібербезпеку у світі за різними сегментами у 2018-2020 роках подана у таблиці 1.6 [9].

Таблиця 1.6-Глобальні витрати на безпеку за сегментами елементів безпеки (млн доларів США)

Сегмент ринку	**2018 рік**	**2019 рік**	**2020 рік**
Захист додатків	2 434	2742	3003
Хмарна безпека	185	304	459
Безпека даних	2563	3063	3524
Управління доступом та автентифікація	8,823	9,768	10 578
Захист інфраструктури	12 583	14 106	15 337

Продовження таблиці 1.6

Сегмент ринку	**2018 рік**	**2019 рік**	**2020 рік**
Комплексне управління ризиками	3949	4 347	4712
Мережева безпека	10 911	12 427	13 321
Інше програмне та апаратне забезпечення для захисту інформації	1 832	2079	2285
Охоронні послуги	52 315	58 920	64 237
Програмне забезпечення для захисту користувачів	5948	6,395	6661
В підсумку	101 544	114 152	124 116

Крім того, все більше компаній вибирають використання систем управління інформацією про безпеку та події (SIEM) як наступного рівня захисту своїх інформаційних систем з метою швидкого виявлення, блокування та розслідування інцидентів. SIEM в комп'ютерній безпеці представляє собою програмні рішення, що об'єднують управління інформаційною безпекою (SIM) та управління подіями безпеки (SEM). Ця технологія забезпечує аналіз подій безпеки в реальному часі, отриманих від мережевих пристроїв та додатків. SIEM включає додатки, пристрої або сервіси та має за мету реєстрацію даних та створення звітів для сумісності з іншими бізнес-даними.

Поняття управління подіями інформаційної безпеки (SIEM) було введено в 2006 році представниками компанії Gartner Марком Ніколлеттом та Амріт Вільямс. В сучасному розумінні SIEM включає:

- Збір, аналіз та передача інформації від мережевих та захисних пристроїв.
- Використання додатків для ідентифікації та управління доступом.
- Застосування інструментів для впровадження політик безпеки та відстеження вразливостей операційних систем, баз даних та журналів додатків.
- Збір інформації про зовнішні загрози.

Зазвичай, SIEM-системи входять до складу оперативних центрів керування кібербезпекою. Наступним етапом управління кібербезпекою є Центр управління безпекою, який об'єднує всю оперативну інформацію про стан безпеки та події, що впливають на кібербезпеку бізнесу. Цей центр також відповідає за прийняття рішень,

формування правил, встановлення процедур та керування всіма заходами, спрямованими на реагування на інциденти або підвищення загальної якості кіберзахисту.

Для компаній Центр операційної безпеки (SOC) виступає як центр компетенції та оперативного ухвалення рішень з усіх питань кібербезпеки. Побудова SOC значно прискорює роботу підрозділів інформаційної безпеки та IT у реагуванні на критичні ситуації, робить її більш злагодженою та ефективною. Інтеграція всієї інфраструктури в єдину платформу управління подіями кібербезпеки, вразливістю, знаннями та інцидентами дозволяє SOC стати постачальником цінної інформації про інформаційні системи та всю інфраструктуру, використовувану IT-підрозділами та бізнес-підрозділами для кращого розуміння того, що відбувається.

З розвитком SIEM-систем набуває популярності концепція систем SOAR (Security Orchestration, Automation and Response). SOAR представляє собою набір сумісних застосунків, які дозволяють організаціям збирати дані про загрози безпеці системи з різних джерел та реагувати на події безпеки низького рівня без втручання людини. Мета використання стека SOAR – підвищення ефективності фізичних та цифрових операцій. Цей термін, введений фірмою Gartner, може відноситися до сумісних продуктів та послуг, які допомагають визначати пріоритети, стандартизувати та автоматизувати функції реагування на інциденти.

Аналіз характеристик SOAR вказує на три ключові можливості цих технологій:

- Управління загрозами та вразливістю: формалізований робочий процес, звітність та комунікаційні можливості для усунення уразливостей.
- Реагування на інциденти: планування, керування, відстеження та координація обробки інцидентів інформаційної безпеки.
- Автоматизація операцій захисту: підтримка автоматизації та оптимізації робочих процесів.

Отже, зростання кіберзлочинності потребує нових рішень захисту інформаційних систем. Сучасні технології, які базуються на інтелектуальному аналізі даних та моделюванні сценаріїв кібератак, надають можливість прогнозувати,

виявляти та блокувати атаки на ранній стадії, що потенційно може значно зменшити збитки та час реагування на інциденти.

1.2 Мережеві атаки

Давайте розглянемо моделі ідентифікації атак хакерів на мережевий трафік, які служать для виявлення та класифікації різних видів атак. Ось декілька з них:

1. **Distributed Denial-of-Service (DDoS) атаки:**

- **Об'ємні атаки (Volumetric Attacks):** Найбільш поширений тип DDoS-атак, що перевантажує пропускну спроможність мережі за допомогою запитів по всіх відкритих портах, таких як UDP-потоки та ICMP-потоки.
- **Атака Smurf або ICMP-флуд:** Використовує широкомовну розсилку для перевірки працюючих вузлів, надсилаючи ping-запити та може призвести до відмови в обслуговуванні.
- **Атака Fraggle (UDP флуд):** Подібна до Smurf-атаки, але використовує пакети UDP, призводячи до насичення смуги пропускання та відмови в обслуговуванні.

2. **Атаки на рівні програми:**

- Спрямовані на прямий веб-трафік, такий як HTTP, HTTPS, DNS або SMTP.
- Ці атаки можуть бути важко виявити, оскільки вони можуть використовувати менше машин.

3. **Інші види атак:**

- **ARP-фальсифікація:** Змінює ARP-таблиці для перенаправлення мережевого трафіку.
- **Виснаження адрес DHCP:** Займає всі доступні IP-адреси, що може викликати відмову в обслуговуванні.
- **Маніпуляції STP:** Змінює протоколи сполучного дерева, спричиняючи збої в мережі.

Це лише огляд, і існує безліч інших видів атак. Важливо вживати заходів для запобігання та виявлення таких атак, щоб забезпечити безпеку мережі та даних.

Об'ємні атаки - це категорія DDoS-атак, спрямованих на перевищення пропускної спроможності мережного каналу, охоплюючи рівні 3, 4 та 7 моделі OSI, що становить приблизно 65% всіх DDoS-атак.

Давайте розглянемо моделі ідентифікації атак хакерів на мережевий трафік, які служать для виявлення та класифікації різних видів атак. Ось декілька з них:

4. **Distributed Denial-of-Service (DDoS) атаки:**

- **Об'ємні атаки (Volumetric Attacks):** Найбільш поширений тип DDoS-атак, що перевантажує пропускну спроможність мережі за допомогою запитів по всіх відкритих портах, таких як UDP-потоки та ICMP-потоки.
- **Атака Smurf або ICMP-флуд:** Використовує широкомовну розсилку для перевірки працюючих вузлів, надсилаючи ping-запити та може призвести до відмови в обслуговуванні.
- **Атака Fraggle (UDP флуд):** Подібна до Smurf-атаки, але використовує пакети UDP, призводячи до насичення смуги пропускання та відмови в обслуговуванні.

5. **Атаки на рівні програми:**

- Спрямовані на прямий веб-трафік, такий як HTTP, HTTPS, DNS або SMTP.
- Ці атаки можуть бути важко виявити, оскільки вони можуть використовувати менше машин.

6. **Інші види атак:**

- **ARP-фальсифікація:** Змінює ARP-таблиці для перенаправлення мережевого трафіку.
- **Виснаження адрес DHCP:** Займає всі доступні IP-адреси, що може викликати відмову в обслуговуванні.
- **Маніпуляції STP:** Змінює протоколи сполучного дерева, спричиняючи збої в мережі.

Це лише огляд, і існує безліч інших видів атак. Важливо вживати заходів для запобігання та виявлення таких атак, щоб забезпечити безпеку мережі та даних.

Об'ємні атаки - це категорія DDoS-атак, спрямованих на перевищення пропускної спроможності мережного каналу, охоплюючи рівні 3, 4 та 7 моделі OSI, що становить приблизно 65% всіх DDoS-атак.

1.3 Технології інтелектуального аналізу подій

Любу подію, що відбувається в інформаційному середовищі, можна виявити за допомогою аналітичних інструментів, зокрема аналізу досліджуваного середовища.

У сучасному світі існують три основні терміни, пов'язані з аналізом даних: штучний інтелект (ШІ), машинне навчання та наука про дані, які часто використовуються як взаємозамінні або взаємопов'язані. Це три різні поняття з такими визначеннями:

- Штучний інтелект (ШІ) - широкий термін, який базується на моделюванні здатностей людини, таких як відчуття, розуміння та реагування, за допомогою комп'ютерів.
- Машинне навчання (МН) - це розділ штучного інтелекту, що використовує статистичні методи для того, щоб надати комп'ютерам можливість "вчитися" під час виконання певного завдання та поступово покращувати свою продуктивність.
- Наука про дані - для реалізації алгоритмів машинного навчання необхідно визначити набори даних, обрати відповідні змінні та показники, а також виконати завдання інформаційної інженерії: пошук прихованих залежностей, збір, підготовку, інтеграцію, візуалізацію даних, визначення продуктивності алгоритму та інше.

Події в галузі кібербезпеки можна ефективно аналізувати з трьох різних точок зору: джерел даних, до яких застосовується аналіз; методів машинного навчання, які використовуються; та прогнозування кінцевих результатів, яких слід досягти.

Основні джерела даних про стан інформаційних систем, їхню безпеку та активність включають журнали подій, журнали обслуговування, аналіз трафіку та

інші. Важливими джерелами даних про події у сфері кібербезпеки є дані користувача, дані програми, дані кінцевої точки та мережеві дані, як показано на рис. 1.3.

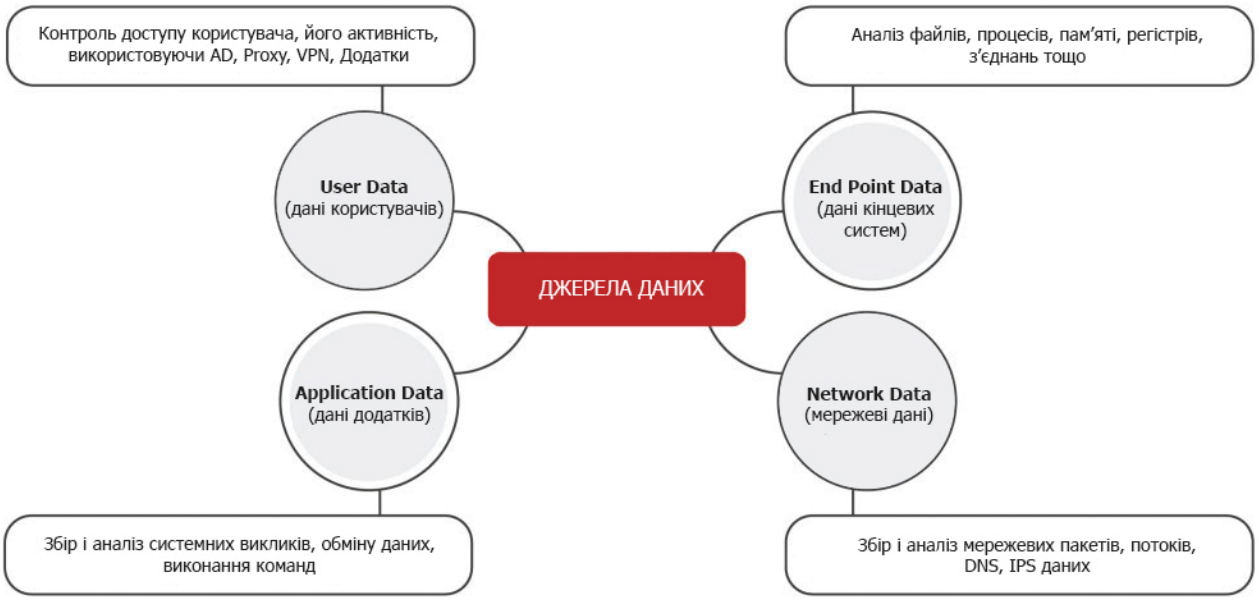

Рисунок 1.2– Джерела даних про події

Наразі існує велика кількість продуктів для збору та аналізу різних видів інформації, таких як аналізатори мережевих пакетів, антивіруси, мережеві екрани, системи IDS/IPS та інші. У даній роботі детально розглядається процес збору та аналізу даних мережевих інтерфейсів і мережевих сервісів.

Звичайне накопичення та аналіз даних діагностичних систем не завжди призводить до бажаних результатів. Іноді виникають ситуації, коли обсяг інформації настільки великий, що знайти значущу інформацію в потрібному контексті стає складним завданням. Це може призвести до того, що постійна зміна характеру кібератак, які можуть суттєво відрізнятись від попередніх, ускладнює прийняття управлінських рішень на основі реагування на події в кіберсистемі. Методи машинного навчання часто використовуються для підвищення швидкості та точності управлінського прийняття рішень в кіберсистемах. З кожним новим набуттям знань якість та швидкість майбутніх аналізів постійно зростають. Приклади методів машинного навчання представлені на рис. 1.4.

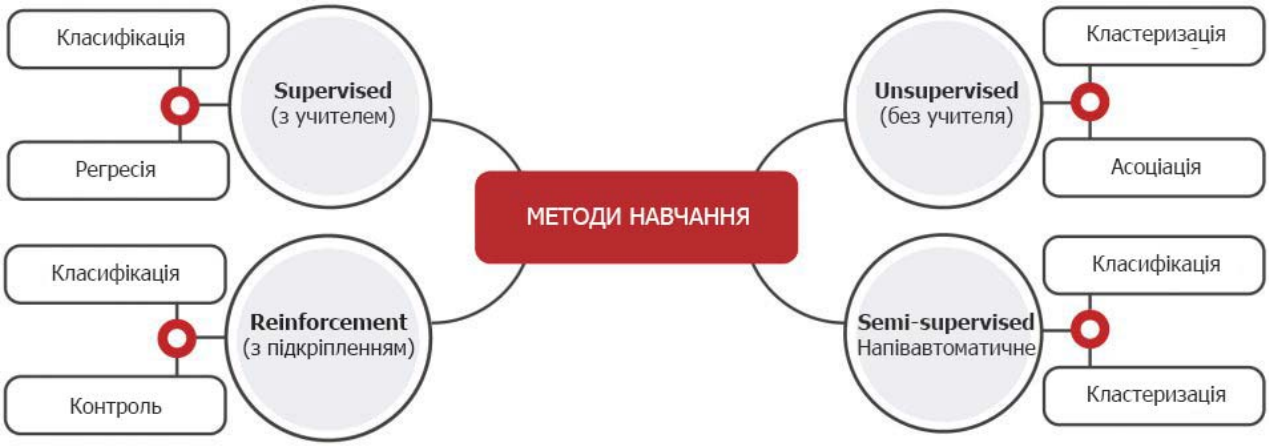

Рисунок 1.3— Методи машинного навчання

1.4 Застосування нейронної мережі

Штучні нейронні мережі, або нейронні мережі, є математичними моделями або їх програмними та апаратними реалізаціями. Ця концепція виникла як відповідь на процеси, що відбуваються у мозку, і спроби змоделювати ці процеси. Основні принципи полягають у тому, щоб інтерпретувати дані від датчиків у вигляді свого роду машинного сприйняття, маркування або групування необроблених даних. Розпізнані цими моделями шаблони є числовими та представлені у векторах, в які перетворюються будь-які інші дані [24].

Кожен вузол в мережі має один або кілька входів та один вихід. Нейрон може працювати у двох режимах: у режимі навчання та у режимі використання або тестування. У режимі навчання нейрон вивчає реагувати на конкретні вхідні шаблони. У робочому режимі нейрон реагує на вхідний шаблон та пов'язує його з вихідним сигналом. Якщо нейрон отримує нетиповий набір параметрів, то він вирішує, чи активувати його.

Кожен вхідний сигнал має свою відповідну вагу, яка розраховується на основі вхідних даних. Якщо отримане число перевищує граничне значення, нейрон спрацьовує. Активацію будь-якого нейрона регулює його функція активації.

У 1999 році Ян Лекун та його колеги створили систему розпізнавання рукописних цифр, яку вони назвали LeNet. Згодом цей новий алгоритм був формалізований як згорткові нейронні мережі (CNN). Головною метою цієї мережі було розпізнавання зображень та відео, аналіз та класифікація зображень, відтворення музики, надання рекомендацій та детальний аналіз людської мови.

У сфері машинного навчання термін "згорткові нейронні мережі" (CNN) вказує на клас штучних нейронних мереж глибокого прямого поширення. Ці мережі відрізняються від інших алгоритмів класифікації меншою потребою в попередній обробці даних.

Архітектура CNN побудована аналогічно схемі з'єднання нейронів людського мозку, що використовується в Visual Cortex. У цій архітектурі певні нейрони реагують лише на стимули, які з'являються в їх рецептивному полі, тобто в обмеженій зоні зорового поля.

Мережа має властивості багатошарового перцептрону, і це відображено в її структурі через використання повнозв'язного шару. Цей шар є простим засобом вивчення нелінійних комбінацій високорівневих функцій, представлених вихідними даними згорткового шару. Зазвичай рівень Fully Connected використовує нелінійну функцію. Після того, як дані інтерпретовані у відповідний формат, вони подаються на вхід багатошаровому перцептрону. Дані надсилаються через мережу прямого зв'язку, і кожною ітерацією навчання використовується зворотнє поширення помилки. За кілька епох модель може відрізняти ключові та низькорівневі функції вхідних даних та класифікувати їх за допомогою методу класифікації Softmax.

На сьогоднішній день існує кілька різних архітектур CNN, які відіграють важливу роль у розвитку штучного інтелекту. До них входять такі, як LeNet, AlexNet, VGGNet, GoogLeNet, ResNet та ZFNet [25].

Рекурентні нейронні мережі (RNN) вважаються однією з потужних архітектур нейронних мереж. RNN - це окремий клас штучних нейронних мереж, який визначається створенням тимчасового графа через зв'язки між вузлами. У рівні RNN використовується цикл for для ітерації впорядкованої за часом послідовності, зберігаючи закодовану інформацію про кроки у внутрішньому стані. Відмінною

рисою RNN є їх здатність обробляти будь-які послідовності вхідних даних, що робить їх ефективними для розв'язання завдань, таких як розпізнавання безперервного несегментованого рукописного тексту та мови.

У 1998 році Зепп Хохрайтер та Юрген Шмідгубер представили архітектуру рекурентних нейронних мереж, зокрема довгострокової пам'яті (LSTM). Мережа LSTM, як і більшість нейронних рекурентних мереж, є універсальною і може обчислити все, що може звичайний комп'ютер за достатню кількість вузлів мережі, якщо є відповідна вагова матриця, що представляє собою її програму. У порівнянні з традиційними RNN, мережі LSTM виявляються зручними для вивчення досвіду та застосування у завданнях класифікації, обробки та прогнозування часових рядів, особливо тих, де існують невизначені затримки між важливими подіями. Завдяки своїй нечутливості до довжини інтервалу мережі LSTM виявляють конкурентні переваги у таких областях, як стиснення тексту природною мовою та розпізнавання рукописного тексту. Досягненням у цьому напрямку була перемога в конкурсі ICDAR з розпізнавання рукописного введення у 2009 році. У 2003 році був встановлений рекорд помилок фонем на рівні 17,7% при використанні мережі LSTM на класичному наборі даних природного мовлення TIMIT для автоматичного розпізнавання мови. На сьогоднішній день провідні компанії, такі як Google, Apple, Microsoft та Baidu, успішно використовують мережі LSTM у своїх нових продуктах [13].

Іншим прикладом простої нейронної мережі є мережа Хопфілда, яка представляє собою рекурентну повністю зв'язкову штучну нейронну мережу з симетричною матрицею зв'язків. Цей тип мережі був створений Джоном Хопфілдом в 1981 році та складається лише з одного шару. Основні завдання, які вирішує ця мережа, включають об'єднання та автоматичну оптимізацію. У відміну від багатьох інших нейронних мереж, які працюють до досягнення необхідної відповіді протягом певної кількості епох, мережі Хопфілда працюють до досягнення стану рівноваги, коли наступний стан мережі ідентичний попередньому [12].

Ще однією стохастичною рекурентною нейронною мережею є машина Больцмана, винайдена Террі Сейновські та Джеффрі Хінтоном у 1986 році. Ця машина може розглядатися як генеративний стохастичний варіант мережі Хопфілда і

отримала свою назву на честь Людвіга Больцмана, австрійського вченого зі статистичної фізики. Машина Больцмана використовує алгоритми моделювання, і, як і мережа Хопфілда, є марківським випадковим полем.

Мережі глибоких переконань (DBN) є ще одним типом глибоких нейронних мереж у машинному навчанні. Вони є генеративними графічними моделями і складаються з кількох шарів прихованих змінних, пов'язаних в межах кожного шару та між шарами. Під час навчання DBN намагається ймовірнісно відновити свої вхідні дані, розглядаючи шари як детектори об'єктів на вхідних даних. Додаткове навчання DBN може бути спрямоване на класифікацію.

Окрім того, DBN можна розглядати як комбінацію простих та спонтанних мереж, таких як обмежені машини Больцмана (RBM) чи автокодувальники. Це сприяє швидкому навчанню, де порівняльна дивергенція застосовується до кожної підмережі, розпочинаючи з "нижніх" пар шарів. Цей підхід був розвинений Йі-Уайє Тенгом, учнем Джеффрі Хінтона, і призвів до одного з перших ефективних алгоритмів машинного навчання [8].

Автоенкодер – це ще одна штучна нейронна мережа, яка базується на неконтрольованому навчанні та використовується для створення генеративних моделей даних. Ця модель шукає залежності між вхідними даними та їх поданням. Архітектурно автоенкодер подібний до багатошарового перцептрону з вхідним і вихідним шарами, але відмінність полягає в тому, що вихідний шар має ту ж саму кількість вузлів, що і вхідний, і навчається відновлювати вхідні дані, замість прогнозування цільового значення. Автоенкодери відносяться до моделей спонтанного навчання [4].

Загалом, кожна з архітектур нейромереж має свої концептуальні переваги з точки зору теоретичного підходу та практичного застосування. Існують і інші архітектури нейронних мереж, які успішно використовуються для різноманітних завдань в житті людини. У даній роботі був використаний багатошаровий перцептрон Румельхарта як модель нейронної мережі.

Багатошаровий перцептрон має функціональні переваги порівняно з персептроном Розенблатта, особливо в тому випадку, коли йому слід реагувати на

подразники з більшою ефективністю. Це можливо через те, що багатошаровий перцептрон може передбачити реакції на різні типи подразників завчасно. Такий підхід сприяє поліпшенню загальної здатності до узагальнення, тобто правильної реакції на стимули, які не були включені до процесу навчання перцептрона. На жаль, наукова література на сьогодні не містить подібних узагальнюючих теорем, і практичні випробування обмежуються лише дослідженням декількох стандартизованих тестів, призначених для порівняння різних архітектур.

Оскільки багатошаровий перцептрон може включати будь-яку кількість шарів, важливо визначити оптимальну кількість шарів для новоствореної моделі. У нашій роботі було використано три шари: вхідний, вихідний та внутрішній. Загалом, запропоновану модель можна представити у форматі, вказаному на рис. 1.5.

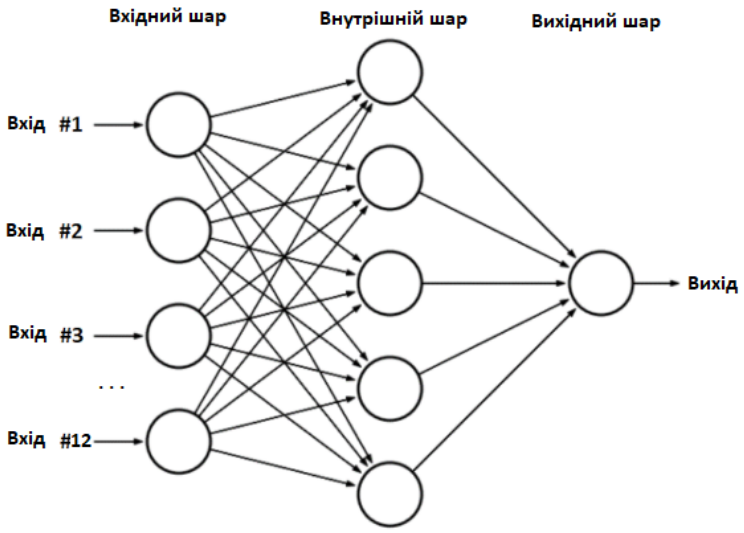

Рисунок 1.5- Простіша нейронної мережі (персептрона).

Багатошаровий перцептрон із зворотнім поширенням помилки в наш час є найширше використовуваною формою багатошарової нейронної мережі, а також вважається одним із найпростіших та загальних методів для контрольованого навчання багатошарових нейронних мереж. Основний принцип зворотного поширення помилки полягає в апроксимації нелінійної залежності між вхідними та вихідними даними шляхом коригування ваги між ними.

Мережа зворотного поширення помилки зазвичай включає два етапи: навчання та тестування. Під час навчання вхідні дані порівнюються з правильними класифікаціями, наприклад, використовуючи закодоване зображення особи та вихідний код, що відповідає імені особи.

Нейронна мережа, так само як і інші алгоритми навчання, повинна перетворювати вхідні та вихідні дані відповідно до зазначеного користувачем шаблону. Архітектура мережі визначається схемою, і після завершення навчання структуру мережі неможливо змінити без створення нової мережі.

Операції мережі зворотного поширення помилки розділяються на два етапи: пряме та зворотне поширення помилки. Під час прямого поширення вхідний шаблон застосовується до вхідного шару, а його вплив розповсюджується через мережу, доки не формується вихідний сигнал. Результат порівнюється з очікуваним виходом, і для кожного вихідного вузла обчислюється сигнал помилки.

Коли сигнали помилок для всіх вузлів визначено, дані про помилки використовуються для оновлення вагових значень кожного з'єднання до тих пір, поки мережа не здатна ефективно кодувати всі навчальні шаблони. Алгоритм зворотного поширення помилки застосовує метод градієнтного спуску для пошуку мінімального значення функції помилок у ваговому просторі. Ваги, що мінімізують функцію помилок, розглядаються як вирішення завдання навчання.

Мережа веде себе аналогічно людині, якій надали набір даних та попросили класифікувати їх за заздалегідь визначеними категоріями. Так само, як і людина, мережа формулює "теорії" щодо того, як шаблони впорядковані в класах. Потім ці теорії перевіряються на відповідність правильним результатам, щоб оцінити точність припущень мережі. З значущими змінами ваг вказується на радикальні зміни у востаннє зазначеній операції, в той час як дрібні зміни можна вважати непомітними коригуваннями.

Також виникає питання про узагальнення нейронної мережі. Ситуації, коли мережа вивчила надто мало чи надто багато, можуть виникнути. Недостатнє навчання може виникнути, якщо нейронна мережа недостатньо складна, щоб виявити закономірність в складному наборі даних. Зазвичай такий випадок є результатом

мереж з недостатньою кількістю прихованих вузлів, які не можуть точно представити рішення через обмеженість даних (див. рис. 1.6).

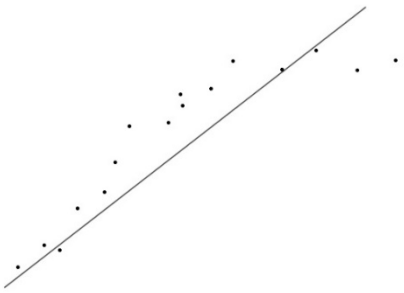

Рисунок 1.6– Крива узагальнення мережі через відсутність прихованих вузлів

Навпаки, перенавчання може надто ускладнити мережу, що призведе до прогнозів, виходячи за межі обсягу навчальних даних. Мережі з великою кількістю прихованих вузлів тенденцію перенадто точно адаптувати рішення (див. рис. 1.7).

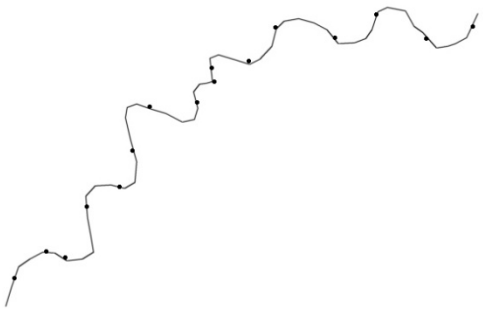

Рисунок 1.7– Крива генералізації мережі через надмірність прихованих вузлів

Таким чином потрібно створити нейронну мережу з «правильною» кількістю прихованих вузлів, що призведе до вирішення завдання узагальнення (рис. 1.8).

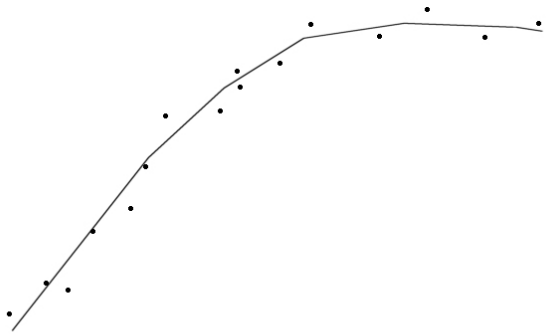

Рисунок 1.8- Крива узагальнення мережі з достатньою кількістю прихованих вузлів

Також потрібно провести навчання мережі.

1.5 Навчання нейронної мережі

Навчання з підкріпленням (Reinforcement Learning) – це метод машинного навчання, при якому модель навчається без вміння про систему, але має можливість взаємодіяти з нею шляхом виконання різних дій. Ці дії переводять систему в новий стан, і модель отримує відповідь від системи у вигляді зворотного зв'язку.

Напівавтоматичне навчання – це тип машинного навчання, що входить у категорію контрольованого навчання, де для тренування використовуються як розмічені, так і нерозмічені дані. Зазвичай використовується невеликий обсяг розмічених даних і значний обсяг нерозмічених. Цей метод займає проміжне положення між навчанням без вчителя (без використання розмічених даних) та навчанням з вчителем (використання лише розмічених даних). Важливо відзначити, що дослідження в галузі машинного навчання показали, що використання нерозмічених даних разом із невеликою кількістю розмічених може значно підвищити точність навчання.

У даній роботі нейромережева модель класифікації подій кібербезпеки ґрунтується на методі тренування викладача. Роль викладача була симульована за допомогою спеціально розробленого програмного забезпечення, яке включає алгоритм обробки вихідних даних і їхнє перетворення у визначений формат, який був описаний у практичній частині роботи.

Кожна модель машинного навчання базується на певному алгоритмі, такому як класифікація, кластеризація, асоціативні правила, глибоке навчання, регресія та зіставлення із зразком. Вибір конкретного алгоритму залежить від кінцевої мети, яку слід досягти. Наприклад, для визначення приналежності даних до певної групи можуть використовуватися інструменти теорії ймовірностей, екземплярні методи, дерева рішень та нейронні мережі. Докладна інформація про кожен із існуючих алгоритмів машинного навчання та їх функціональність наведена на рис. 1.9.

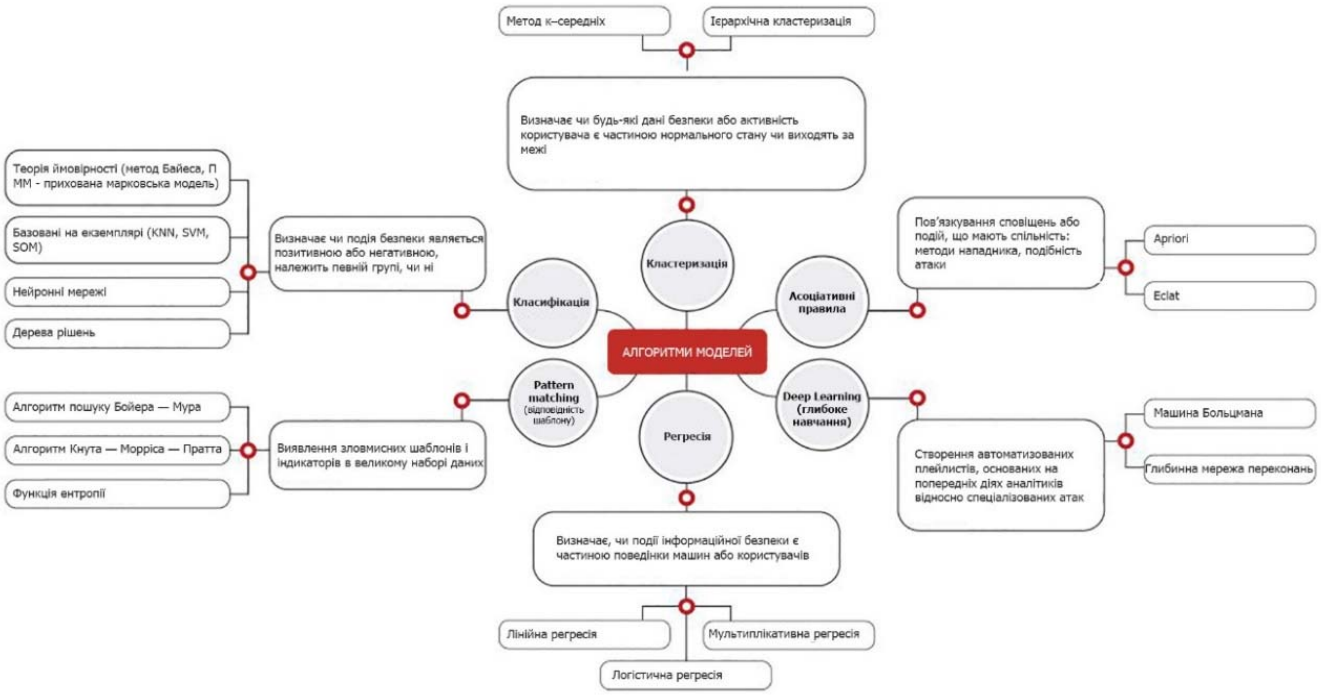

Рисунок 1.9. – Алгоритми моделей машинного навчання

Розглянемо приклади використання зазначених моделей. Байєсовський метод, заснований на теорії ймовірностей та статистичному аналізі, застосовується для створення спам-фільтрів та виявлення фішингу, класифікуючи звичайні та спам-повідомлення.

Для виявлення шахрайства в Інтернеті використовуються нейронні мережі та системи підтримки ухвалення рішень щодо шахрайства.

Методи кластеризації використовуються для виявлення внутрішніх загроз, таких як порушення конфіденційності користувачів або витік даних.

Ботів можна виявляти за допомогою ентропійних функцій, які застосовуються до результатів міжмашинної взаємодії.

Асоціативний аналіз дозволяє виявляти групи зловмисників, які використовують примітивні (відомі) методи атак у мережі.

Успішне управління бізнесом вимагає вміння прогнозувати можливі події безпеки, ефективно боротися з погрозами та аналізувати всі ризики. Це представляє найвищий рівень в побудові інтелектуальних систем, і тому використовуються різноманітні методи аналізу, які представлені на рис. 1.10.

Рисунок 1.10 - Моделі аналізу подій кібербезпеки.

Описовий аналіз, або історичний огляд, є цінним, оскільки дозволяє отримати уроки з попередніх моделей поведінки та зрозуміти можливий вплив на майбутні результати.

Розумна аналітика, також відома як предиктивна аналітика, спроможна оцінювати ймовірність майбутніх результатів. Важливо враховувати, що жоден статистичний алгоритм не може передбачити майбутнє зі 100% впевненістю.

Компанії використовують цю статистику, щоб передбачити події, які можуть статися у майбутньому.

Наказує аналітика дозволяє користувачам визначити різні можливі дії і направляти їх для вирішення проблем. Вона кількісно оцінює вплив майбутніх рішень, надаючи рекомендації щодо можливих результатів до прийняття рішень.

Діагностична аналітика є формою розширеної аналітики, яка досліджує дані чи контент, щоб відповісти на питання "Чому це сталося?"

Детективна аналітика заснована на аналізі та виявленні невідомих об'єктів, які можуть становити певну загрозу.

В інформаційній безпеці немає універсального методу захисту від усіх загроз. Схоже на те, що із зростанням загроз інформаційній безпеці потрібен різноманітний підхід, який охоплює різні аспекти та види аналітики.

У таблиці 1.7 показано відповідність аналітичної системи основним джерелам загроз.

Таблиця 1.7 – Актуальність аналітичної системи до основних джерел загроз

Джерело загрози	Мережевий аналіз	Аналіз поведінки користувачів	Кінцевий системний аналіз	Аналіз додатків
Розширене шкідливе ПЗ	✓	✗	✓	✗
соціальна інженерія	✓	✓	✓	✗
«Бічний рух у мережі»	✓	✓	✓	✗
Інсайдери	✗	✓	✗	✗
Шахрайство під час транзакцій	✗	✗	✗	✓
Отримання облікових записів від іншого користувача	✗	✓	✗	✗
Видалення даних	✓	✓	✗	✓
Запуск експлойтів	✗	✗	✗	✓
Зашифровані атаки	✗	✗	✓	✓

Сучасна аналітична система має включати кілька спеціалізованих аналітичних систем та об'єднувати результати обробки кожної з них.

Взагалі, комплексна модель для прийняття рішень щодо конкретної події в інформаційній системі повинна охоплювати всі три концепції, які були розглянуті вище. Отже, комплексна модель характеризується такими аспектами:

- Повинна мати постійне джерело знань про події (логи, інформація про трафік та інше).
- На основі неперервного накопичення даних повинна надавати результати їх обробки з постійним вдосконаленням цих результатів.
- Здійснювати остаточний аналіз та повідомляти аналітика.

Базову модель побудови інтелектуальної системи виявлення та ідентифікації подій кібербезпеки представлено на рис. 1.11.

Рисунок 1.11– Базова модель побудови інтелектуальної системи виявлення подій.

Отже, проведення інтелектуального аналізу подій в галузі кібербезпеки включає в себе етапи ідентифікації джерела даних, їх аналізу, подальшого використання машинного навчання та завершального аналізу. Це є фундаментальною моделлю для створення такої інтелектуальної системи.

1.6 Завдання дослідження

Об'єктом цього дослідження є класифікація подій у сфері кібербезпеки за допомогою інтелектуальних моделей. Зважаючи на обширний обсяг завдання, яке включає в себе комплексну серію дій (засновану на попередньому аналізі) і орієнтоване на постійне вдосконалення, метою даної дипломної роботи є розробка ефективної моделі виявлення та ідентифікації кібербезпеки подій, спрямованої на конкретні загрози мережі. Це включає розробку програмного забезпечення, чітке визначення ключових етапів або використання наявної нейронної мережі.

З метою досягнення поставлених завдань були визначені наступні кроки:

- Збір інформації про наявні мережеві атаки, включаючи акумуляцію даних мережевого трафіку та журналів подій.
- Структурування даних в узагальнений формат, організація їх у мережеві потоки даних.
- Розробка алгоритмів підготовки даних.
- Розробка та навчання нейромережевих моделей.
- Перевірка адекватності синтезованих моделей.

Кожен вид мережевих атак має свої особливості та технологічні відмінності. У цьому дослідженні було розглянуто кілька моделей нейромереж та проаналізовано їхню придатність для комплексного вивчення.

Висновки по першому розділу

У першому розділі висвітлено ключові глобальні тенденції у розвитку кібербезпеки, що мають місце з початку 10-х років XXI століття. З метою поліпшення якості кібербезпеки надточас виявляється збільшене використання засобів інтелектуального аналізу даних у сфері комп'ютеризації. Різноманітні алгоритми машинного навчання відкривають широкий спектр гнучких заходів для протидії кібератакам. Наступні розділи детально описують розробку деяких з цих моделей.

РОЗДІЛ 2 РОЗРОБКА МОДЕЛІ ІДЕНТИФІКАЦІЇ АТАКИ НА МЕРЕЖЕВИЙ ТРАФІК

2.1. Мережеві атаки

Розглянемо моделі ідентифікації атак хакерів на мережевий трафік. Ці моделі допомагають виявити та класифікувати різні види атак. Ось деякі з них:

1. Distributed Denial-of-Service (DDoS) атаки:

o Об'ємні атаки (Volumetric Attacks): Це найбільш поширений тип DDoS-атак Він перевантажує пропускну спроможність мережі, засмічуючи її помилковими запитами по всіх відкритих портах. UDP-потоки та ICMP-потоки є основними формами об'ємних атак.

o Атака Smurf або ICMP-флуд: Зловмисник використовує широкомовну розсилку для перевірки працюючих вузлів у системі, надсилаючи ping-запити. Це може призвести до відмови в обслуговуванні жертви.

o Атака Fraggle (UDP флуд): Це аналогічно Smurf-атаці, але використовує пакети UDP Вона також призводить до насичення смуги пропускання та відмови в обслуговуванні.

2. Атаки на рівні програми:

o На цьому рівні фокусуються на прямому веб-трафіку. Можливі варіанти включають HTTP, HTTPS, DNS чи SMTP.

o Ці атаки не завжди легко виявити, оскільки вони можуть використовувати менше машин.

3. Інші види атак:

o ARP-фальсифікація: Зловмисник змінює ARP-таблиці, щоб перенаправити мережевий трафік.

o Виснаження адрес DHCP: Зловмисник займає всі доступні IP-адреси, що може призвести до відмови в обслуговуванні.

- Маніпуляції STP: Зловмисник може змінювати протоколи сполучного дерева, що може призвести до збоїв у мережі.

Це лише невеликий огляд і існує безліч інших видів атак. Важливо вживати заходів для запобігання та виявлення таких атак, щоб забезпечити безпеку мережі та даних.

Об'ємні атаки (Volumetric Attacks)- Це одна з категорій DDoS-атак, яка спрямована на перевищення пропускної спроможності мережного каналу. Ці атаки охоплюють рівні 3, 4 та 7 моделі OSI і становлять близько 65% всіх DDoS-атак.

Як, що розглядати докладніше, що таке об'ємні атаки то:

1. Рівень програми (L7):

- На цьому рівні забезпечується взаємодія додатків користувача з мережею. Приклади включають перегляд веб-сторінок через протокол HTTP.
- Об'ємні атаки на рівні програми можуть перевантажити сервери, надсилаючи багато запитів на конкретні ресурси.

2. Рівень вистави (L6):

- Цей рівень забезпечує перетворення протоколів та кодування/декодування даних. Він використовує протоколи стиснення та кодування даних (наприклад, ASCII, EBCDIC).

3. Транспортний рівень (L4):

- L4 забезпечує надійну передачу даних від відправника до отримувача. Основні протоколи цього рівня – UDP та TCP.
- Об'ємні атаки на L4 можуть перевантажити пропускну здатність каналу, викликаючи відмову в обслуговуванні.

4. Мережевий рівень (L3):

- L3 відповідає за трансляцію логічних адрес та імен, комутацію та маршрутизацію. Працює за протоколом IP (Internet Protocol).
- Атаки на L3 можуть перевантажити маршрутизатори та мережні пристрої.

5. Канальний рівень (L2):

- L2 забезпечує взаємодію мереж фізично через комутатори та концентратори.

6. Фізичний рівень (L1):

o L1 визначає спосіб передачі даних між пристроями. Працює завдяки протоколам Ethernet, Bluetooth, Wi-Fi та IRDA.

Об'ємні DDoS-атаки вимірюються в бітах за секунду (біт/с). Зловмисники прагнуть перевантажити пропускну спроможність мережі, відправляючи величезну кількість трафіку чи запитів на мету. Це може викликати відмову в обслуговуванні та серйозні проблеми для організації.

Приклади об'ємних атак включають атаки на рівні протоколу (наприклад, UDP або TCP флуд) і атаки на рівні додатків (наприклад, HTTP флуд). Важливо мати ефективні заходи захисту, щоб мінімізувати вплив таких атак на мережевий трафік.

Атаки на рівні програми— це одна з найпоширеніших і найнебезпечніших форм атак на мережевий трафік. Вони спрямовані на перевантаження ресурсів сервера або програми, таких як процесорний час, пам'ять або база даних. Давайте розглянемо їх докладніше:

1. Рівень програми (L7):

o На цьому рівні забезпечується взаємодія додатків користувача з мережею. Приклади включають перегляд веб-сторінок через протокол HTTP.

o DDoS-атаки рівня 7 (L7)відносяться до типу шкідливих програм, призначених для «верхнього» рівня моделі OSI, де відбуваються поширені інтернет-запити, такі як HTTP GET і HTTP POST2.

o Атаки на рівні програми можуть перевантажити сервери, надсилаючи велику кількість запитів на конкретні ресурси.

2. Приклади атак на рівні програми:

o SQL-ін'єкції: Зловмисник впроваджує шкідливий SQL-код у запити до бази даних, що може призвести до витоку або зміни даних.

o Cross-Site Scripting (XSS): Зловмисник впроваджує шкідливий скрипт на веб-сторінку, що виконується на стороні клієнта, коли користувач її відвідує.

o Path Traversal: Зловмисник намагається отримати доступ до файлів або директорій, до яких він не має дозволу, шляхом маніпуляції шляхами URL.

3. Захист від атак на рівні програми:

- Міжмережні екрани прикладного рівня (WAF): WAF виявляє та запобігає відомим атакам на рівні програми та бізнес-логіки. Він також виявляє експлуатацію вразливостей нульового дня та аналізує події для виявлення ланцюжків атак1.
- Оновлення та гарне кодування: Регулярне оновлення програм та використання безпечних методів кодування допоможуть запобігти атакам.

Атаки на рівні програми вимагають уваги та захисту, щоб забезпечити безпеку веб-додатків та захистити користувачів від витоку даних та інших загроз.

Інші види атак на мережевий трафік

На додаток до об'ємних атак та атак на рівні програми, існує ряд інших видів атак, які можуть загрожувати безпеці мереж. Давайте розглянемо деякі з них:

1. IP Null атака:
   - Зловмисники встановлюють значення поля "Protocol" у заголовку IP-пакета рівним нулю.
   - Це дозволяє відправляти пакети у великій кількості, обминаючи фаєрволи та маршрутизатори.
   - Системні ресурси жертви зайняті аналізом трафіку, що може призвести до відмови в обслуговуванні.
2. SYN-флуд:
   - Зловмисник надсилає запити на з'єднання з сервером із підробленими IP-адресами джерела.
   - Це перевантажує таблицю з'єднань сервера, викликаючи відмову в обслуговуванні.
3. UDP-флуд:
   - Зловмисник відправляє велику кількість UDP-пакетів від різних IP-адрес.
   - Переповнення мережного обладнання призводить до перевантаження інтерфейсів та відмови в обслуговуванні.
4. Ping of Death:
   - Зловмисник генерує неправильно сформовані чи надто великі пакети за допомогою команди ping.

- При перевищенні стандартного розміру пакета (65535 байт) виникають збої через переповнення пам'яті.

5. ARP-фальсифікація:

- Зловмисник змінює ARP-таблиці, щоб перенаправити мережевий трафік.

6. Виснаження адрес DHCP:

- Зловмисник займає всі доступні IP-адреси, що може призвести до відмови в обслуговуванні.

7. Маніпуляції STP:

- Зловмисник змінює протоколи сполучного дерева, викликаючи збої в мережі.

Це лише невеликий огляд інших видів атак. Важливо вживати заходів для запобігання та виявлення таких атак, щоб забезпечити безпеку мереж та даних.

2.2. Нейронні мережі (персептрон)

Штучні нейронні мережі (нейронні мережі) — це математичні моделі, і навіть їх програмні чи апаратні реалізації. Ця концепція виникла щодо процесів, які у мозку, і за спробах змоделювати ці процеси. Основними принципами є інтерпретація даних датчиків у вигляді свого роду машинного сприйняття, маркування чи угруповання необроблених даних. Шаблони, що розпізнаються ними, є числовими і містяться у векторах, в які перетворюються будь-які інші дані [24].

Кожен вузол має один або кілька входів та один вихід. Нейрон має два режими роботи: режим навчання та режим використання або тестування. У режимі навчання нейрон навчається реагувати на певні вхідні шаблони. У робочому режимі нейрон реагує вхідний шаблон і пов'язує вихідний сигнал. Якщо нейрон отримує на вхід нетиповий набір параметрів, то він визначає, активувати його чи ні.

Кожен вхідний сигнал має відповідну вагу, яка розраховується на основі вхідних даних. Якщо це число перевищить граничне значення, нейрон спрацює.

Активація будь-якого нейрона регулюється його функцією активації.

2.3. Вибір нейронної мережі для обробки даних

Нейронна мережа— це математична модель, і навіть її програмне чи апаратне втілення, побудоване за принципом організації та функціонування біологічних нейронних мереж. Давайте розглянемо структуру нейронної мережі докладніше:

1. Нейрони (або вузли):

- Нейронна мережа складається з безлічі взаємозалежних нейронів.
- Кожен нейрон приймає вхідні сигнали, обробляє їх та передає результат далі.

2. Шари:

- Нейрони групуються у шари.
- Основні типи шарів:

- Вхідний шар: Приймає вхідні дані.
- Приховані шари (якщо є): Обробляють дані між вхідним та вихідним шарами.
- Вихідний шар: Генерує остаточний результат

3. Зв'язки (ваги):

- Кожен зв'язок між нейронами має вагу.
- Ваги визначають важливість вхідних сигналів для виходу.

4. Функції активації:

- Нейрони використовують функції активації прийняття рішень.
- Приклади функції активації: сигмоїд, ReLU, гіперболічний тангенс.

5. Навчання:

- Нейронні мережі навчаються з урахуванням навчальних даних.
- Методи навчання включають зворотне поширення помилки, градієнтний спуск та інші.

6. Архітектури нейронних мереж:

- Простий перцептрон: Один вхідний шар, один вихідний шар
- Багатошаровий перцептрон (MLP): Містить приховані шари між вхідним та вихідним шарами.

- Згорткові нейронні мережі (CNN): Використовується для обробки зображень.
- Рекурентні нейронні мережі (RNN): Використовується для послідовних даних.

Нейронні мережі мають здатність навчатися, виявляти складні залежності та виконувати узагальнення. Вони знаходять застосування в прогнозуванні, розпізнаванні образів, управлінні та інших областях.

До першого покоління нейронних мереж належать перцептрони. Під персептроном, чи персептроном, розуміють відповідну математичну чи обчислювальну модель, з допомогою якої мозок сприймає інформацію (іншими словами, кібернетичну модель мозку). Вперше ця концепція з'явилася 1955 року у роботі Френка Розенблатта «Перцептрон: імовірнісна модель зберігання та організації інформації у мозку». У науковій літературі персептрон часто згадується під назвою нейронна мережа прямого поширення. Хоча персептрон відносно простий, він навчений вирішувати дуже складні завдання. Зауважимо, що основне математичне завдання, для вирішення якої його можна використовувати, — забезпечення лінійної роздільності, тобто лінійного поділу довільних нелінійних множин. З огляду на те, що мережа має достатню кількість прихованих нейронів, вона завжди може теоретично змоделювати взаємозв'язок між входом і виходом. На практиці використання перцептронів дуже обмежене, але їх часто використовують у поєднанні з іншими мережами для формування нових мереж.

Перцептрон— це математична чи комп'ютерна модель сприйняття інформації мозком, запропонована Френком Розенблаттом у 1957 році. Він є однією з перших моделей штучних нейронних мереж та був реалізований у вигляді електронної машини «Марк-1» у 1960 році1.

Ось основні поняття та архітектурні деталі перцептрону:

1. Структура перцептрону:

- Перцептрон складається із трьох типів елементів:

- Вхідні нейрони: Отримують сигнали від датчиків або інших джерел.

- Приховані нейрони (якщо є): Обробляють вхідні сигнали та передають їх далі
- Вихідні нейрони: Генерують остаточний результат

- У біологічному плані перцептрони аналогічні перетворення зорової інформації на фізіологічну відповідь від рухових нейронів.

2. Принцип роботи:

- Перцептрони створюють набір асоціацій між вхідними стимулами та необхідною реакцією на виході.
- Вони використовують граничну передатну функцію для прийняття рішень.
- Пряме поширення сигналу - вхідні дані передаються через приховані шари вихідним нейронам.

Принцип роботи перцептрону наступний:

1. Вхідні нейрони:

- На вході перцептрон знаходиться набір вхідних сигналів.
- Кожен вхідний нейрон приймає один бінарний вхід (1 або 0).

2. Зв'язки та ваги:

- Зв'язки між вхідними та вихідними нейронами мають свої ваги.
- Ваги визначають значення кожного параметра для обчислення значення.

3. Підсумовування та функція активації:

- Зважені суми вхідних сигналів передаються вихідний нейрон.
- Застосовується функція активації (наприклад, гранична функція) для прийняття рішення.

4. Вихідний нейрон генерує остаточний результат.

5. Навчання.

- Перцептрон навчається з урахуванням навчальних даних.
- Метод зворотного поширення помилки коригує ваги нейронів.

o **Застосування** у прогнозування, розпізнавання образів, керування агентами.

Перцептрони можуть вирішувати прості завдання, але обмежені у вирішенні лінійно нероздільних задач. Їхні принципи лягли в основу складніших архітектур нейронних мереж

Перцептрони можуть вирішувати прості завдання, але обмежені у вирішенні лінійно нероздільних задач. Їхні принципи лягли в основу складніших архітектур нейронних мереж.

У 1999 році Ян Лекун та його колеги розробили розпізнавач рукописних цифр під назвою LeNet. Пізніше новостворений алгоритм був формалізований під назвою згорткові (згорткові) нейронні мережі (CNN). Основним завданням цієї мережі було розпізнавання зображень та відео, аналіз та класифікація зображень, відтворення музики, видача рекомендацій та детальний аналіз людської мови.

У машинному навчанні під нейронними мережами (Convolutional Neural Networks) розуміють клас штучних нейронних мереж глибокого прямого поширення. Потреба попередньої обробки даних для цієї мережі набагато менша в порівнянні з іншими алгоритмами класифікації.

Архітектура CNN побудована аналогічно до схеми з'єднання нейронів людського мозку, використовуваної організацією Visual Cortex. У цьому певні нейрони реагують будь-які стимули лише у рецептивному полі, тобто у обмеженому ділянці зорового поля.

Мережа має властивості багатошарового перцептрону. Його інтерпретація в CNN відображається у вигляді додавання повнозв'язного шару (повнозв'язкового шару). Це (як правило) простий спосіб вивчення нелінійних комбінацій функцій високого рівня, представлених вихідними даними згорткового шару. Рівень Fully Connected, як правило, використовує нелінійну функцію. Коли дані інтерпретуються у відповідний формат, подаються на вхід багатошарового перцептрону. Дані подаються до мережі прямого зв'язку, і кожної ітерації навчання застосовується зворотне поширення помилки. Протягом ряду епох модель здатна розрізняти домінуючі функції та деякі низькорівневі функції вхідних даних та класифікувати їх за допомогою методу класифікації Softmax.

Сьогодні існує кілька різних архітектур CNN, які мають основне значення для розвитку штучного інтелекту в найближчому майбутньому, до них належать: LeNet, AlexNet, VGGNet, GoogLeNet, ResNet, ZFNet [25].

Рекурентні нейронні мережі (RNN) - одна з найпотужніших архітектур нейронних мереж. RNN - окремий клас штучних нейронних мереж, що характеризується формуванням тимчасового графа через зв'язки між вузлами. Якщо схематично представити, рівень RNN використовує цикл for для ітерації впорядкованої за часом послідовності, зберігаючи при цьому закодовану інформацію про кроки у внутрішньому стані, яке він вже бачив. На відміну від нейронних мереж прямого поширення, RNN використовуються для обробки довільних послідовностей вхідних даних, що, у свою чергу, дозволяє застосовувати їх при вирішенні таких завдань, як розпізнавання безперервного несегментованого рукописного тексту та мови.

У 1998 році Зепп Хохрайтер та Юрген Шмідгубер запропонували архітектуру рекурентних нейронних мереж, а саме довгострокової пам'яті (LSTM). Мережа LSTM, як і більшість нейронних рекурентних мереж, вважається універсальною, оскільки вона може обчислити все, що звичайний комп'ютер може обчислити у разі достатньої кількості вузлів мережі, за умови, що вона має відповідну вагову матрицю, яку можна розглядати як ваша програма. З іншого боку, на відміну від традиційних RNN, мережу LSTM зручно використовуватиме вивчення досвіду з метою класифікації, обробки чи прогнозування часових рядів, коли між важливими подіями виникають затримки невизначеної тривалості. Враховуючи, що мережі LSTM відносно нечутливі до довжини інтервалу, вони мають конкурентні переваги перед альтернативними нейронними рекурентними мережами, прихованими марківськими моделями та іншими методами навчання послідовностей у різних додатках. Інші приклади успішного використання мереж LSTM включають найбільш відомі результати зі стиснення тексту природною мовою та безперервного розпізнавання несегментованого рукописного тексту. Підтвердженням цього є перемога у конкурсі ICDAR з розпізнавання рукописного введення у 2009 році. У 2003 році було встановлено рекордний рівень помилок фонем у 17,7% при використанні мережі

LSTM на класичному наборі даних природного мовлення TIMIT при використанні її в автоматичному розпізнаванні мови. Сьогодні провідні компанії, у тому числі Google, Apple, Microsoft та Baidu, використовують мережі LSTM у складі своїх нових продуктів [13].

Ще одна дуже проста нейронна мережа – мережа Хопфілда – це тип рекурентної, повністю зв'язкової штучної нейронної мережі з симетричною матрицею зв'язків. Авторство такої мережі, що має лише один шар, належить Джону Хопфілду і датується 1981 роком. Основні завдання, для яких використовується така мережа, можна визначити як об'єднання та автоматичну оптимізацію. На відміну від багатьох нейронних мереж, які працюють доти, доки через певну кількість епох не буде отримана необхідна відповідь, мережі Хопфілда працюють доти, доки не буде досягнуто рівноваги, коли наступний стан мережі буде таким самим, як і попередній [12]].

Окремим типом стохастичної рекурентної нейронної мережі є машина Больцмана, винайдена Террі Сейновськи та Джеффрі Хінтоном у 1986 році та названа на честь одного з авторів статистичної фізики, австрійського вченого Людвіга Больцмана. Машину Больцмана можна розглядати як генеративний стохастичний варіант мережі Хопфілда, яка також відома в статистиці як марківське випадкове поле.

Робота такої мережі у тому, що з навчання використовуються алгоритми моделювання. Поруч із машину Больцмана можна назвати першої нейронної мережею, здатної навчатися у вигляді внутрішнього уявлення та вирішувати складні комбінаторні завдання. Однак поряд з цим через низку проблем з необмеженою зв'язністю машина Больцмана не може бути використана для вирішення конкретних завдань на практиці. З іншого боку, якщо можливості підключення обмежені, навчання може бути ефективним для практичного застосування. Зокрема так звана мережа глибокої довіри будується з каскаду обмежених машин Больцмана [2].

Інший тип глибокої нейронної мережі в машинному навчанні можна назвати мережею глибоких переконань (DBN), яка є генеративною графічною моделлю. Така мережа складається з кількох шарів прихованих змінних («прихованих вузлів»), які пов'язані між вузлами всередині кожного шару, а між шарами.

При навчанні на наборі прикладів мережа глибоких переконань навчається реконструювати свої вхідні дані у ймовірнісний спосіб. І тут шари визначаються як детектори об'єктів на вхідних даних. Додаткове навчання ДБН після проходження етапу навчання може продовжуватись контрольованим чином для виконання класифікації.

Крім того, DBN також можна розглядати як композицію простих та спонтанних мереж, зокрема обмежених машин Больцмана (RBM) або автокодувальників, що характеризуються тим, що прихований рівень кожної підмережі діє як видимий рівень для наступної. Це також призводить до швидкої спонтанної процедури пошарового навчання, в якій порівняльна дивергенція застосовується до кожної підмережі, починаючи з «нижньої» пари шарів (де найнижчим видимим шаром є навчальний набір).

Один із перших ефективних алгоритмів машинного навчання був розроблений в результаті спостереження, що DBN може навчатися шар за шаром Йі-Уайє Тех, учнем Джеффрі Хінтона [8].

Автоенкодер - це штучна нейронна мережа, заснована на неконтрольованому навчанні та використовується для навчання генеративних моделей даних. Парадигма, що використовується цією мережею, - пошук залежностей між вхідними даними та їх поданням.

З архітектурної точки зору найпростіша форма автокодувальника - це нерекурентна нейронна мережа з прямим зв'язком, яка дуже схожа на багатошаровий перцептрон (MLP) з вхідним шаром, вихідним шаром і одним або декількома прихованими шарами, що їх з'єднують. Проте відмінності між автокодировщиками і MLP полягають у тому, що у автокодировщике вихідний шар має таку кількість вузлів, як і вхідний шар, і замість навчання прогнозованим цільовому значенню Y для заданих вхідних даних відновлюються ваші власні записи. X. Таким чином, автоенкодери є моделі спонтанного навчання [4].

Підсумовуючи вищесказане, можна дійти невтішного висновку, кожна з архітектур нейромереж містить свої концептуальні переваги над іншими, що з теоретичним підходом до проекту і практичної значимістю. Існують і інші

архітектури нейронних мереж, які використовуються для вирішення різних завдань у житті людини. Основним завданням розробленої моделі є класифікація атак, тому мережа має бути класифікатором заданого набору даних. Перцептрони є найпростішими для завдань класифікації у сфері нейронних мереж, у роботі була використана одна з їх концепцій.

Синтез моделі нейронної мережі здійснювався на основі багатошарового перцептрону Румельхарта, тобто окремого випадку перцептрону Розенблатта, що характеризується тим, що корекція вагових коефіцієнтів нейронів відбувається методом зворотного поширення помилки. алгоритм помилки. Фактично наявність більш як одного шару визначається як його характеристика. Зазвичай розглядають два чи три шари [24].

Для кращого розуміння особливостей цих типів перцептронів необхідно зазначити основні відмінності між багатошаровим перцептроном і персептроном Розенблатта:

- використовується нелінійна функція активації, зазвичай сигмоподібна;
- доступний більше одного шару (зазвичай використовується трохи більше трьох);
- сигнали, що надходять на вхід і одержувані з виходу, не є двійковими, а можуть бути закодовані десятковими числами, які необхідно нормалізувати так, щоб значення сегмента були від 0 до 1 (нормалізація необхідна як мінімум для вихідних даних , тому за функцією активації - сигмовидна);
- допускається довільна архітектура підключення (у тому числі повні мережі);
- помилка мережі розраховується не як кількість неправильних зображень після ітерації навчання, а як деякий статистичний захід невідповідності між шуканим та отриманим значенням;
- навчання проводиться не доти, доки не буде помилок після навчання, а доти, доки вагові коефіцієнти не стабілізуються в ході навчання або воно не буде зупинено раніше, щоб уникнути перенавчання.

Функціональні переваги багатошарового перцептрону в порівнянні з персептроном Розенблатта будуть спостерігатися в тому випадку, коли у відповідь на подразники реакції будуть проводитися з більшою ефективністю (бо перцептрон може заздалегідь отримати по одній реакції кожного типу). Таким чином, це призведе до покращення здатності до узагальнення, тобто правильного реагування на стимули, до яких перцептрон не був навчений. Однак на сьогоднішній день подібних узагальнюючих теорем у науковій літературі немає, а на практиці є лише дослідження кількох стандартизованих тестів, які використовуються для порівняння різних архітектур.

Оскільки багатошаровий перцептрон може містити довільну кількість шарів, необхідно визначити оптимальну кількість шарів для новоствореної моделі.

У роботі використовувалося 3 шари: вхідний, вихідний та внутрішній шар. Загалом запропоновану модель можна подати у форматі, показаному рис. 2.1.

2.4. Опис архітектури нейронної мережі

Найперше, під час процесу навчання нейронної мережі виконується операція прямого поширення. Під час навчання, коли конкретний шаблон подається на вхідний шар, вхідна сума *j*-го вузла прихованого шару визначається за допомогою формули 2.1.

$$Net_j = \sum w_{ij} x_j + \theta_j \tag{2.1}$$

Рівняння 2.1 використовується для обчислення загального вхідного сигналу нейрона, який представляє собою взважене значення вузла зміщення. Зазначимо, що вузол зміщення є "псевдовходом" для кожного нейрона в прихованому та вихідному шарах, завжди маючи вихідне значення, що дорівнює 1. Використання вузла зміщення важливе для вирішення ситуацій, коли вхідний сигнал за замовчуванням рівний нулю. У випадках, коли вхідний шаблон містить нульові значення,

використання вузла зміщення необхідне для ефективного навчання нейронної мережі θ_j.

Для вирішення питання активації нейрона використовується значення потенціалу дії, яке передається функції активації. Отримане значення функції активації визначає вихід нейрона і виступає в якості вхідного значення для наступних шарів нейронів, пов'язаних з цим Net_j.

Оскільки алгоритм зворотного поширення помилки вимагає диференціювання функції активації, часто використовується функція сигмоїди.

$$O_j = x_k = \frac{1}{1 + e^{-Net_j}} \tag{2.2}$$

Важливо відзначити, що інші типи функцій також можуть бути використані, наприклад, гіперболічні. Рівняння 2.1 та 2.2 застосовуються для визначення вихідного значення вузла *k* у вихідному шарі.

Далі, завданням є обчислення похибки ваги та її корекція, процес якого відомий як зворотне поширення помилки. Розглянемо операції, які відбуваються на вихідному шарі.

Якщо фактичне значення активації вихідного вузла k дорівнює O_k, а очікуване цільове значення вузла k дорівнює t_k, то різниця між фактичним та очікуваним виходом визначається виразом: $O_k t_k$.

$$\Delta_k = t_k - O_k \tag{2.3}$$

Сигнал помилки для вузла *k* у вихідному шарі можна обчислити за допомогою наступної формули:

$$\delta_k = \Delta_k O_k (1 - O_k) \tag{2.4}$$

або

$$\delta_k = (t_k - O_k)O_k(1 - O_k) \quad (2,5)$$

де - похідна сигмовидної функції.$O_k(1 - O_k)$

Відповідно до правила дельта, зміна ваги між вхідним вузлом *j* та вихідним вузлом *k* пропорційна помилці в вузлі *k*, помноженій на активацію вузла *j*. Формули 2.6 та 2.7 використовуються для коригування ваги між вихідним вузлом *k* та вузлом *j*. $w_{j,k}$.

$$\Delta w_{j,k} = l_r \delta_k x_k \quad (2,6)$$

$$w_{j,k} = w_{j,k} + \Delta w_{j,k} \quad (2,7)$$

де зміна ваги між вузлами jek, $\Delta w_{j,k}$

l_r- швидкість навчання.

Коефіцієнт навчання - це стала, що вказує на те, наскільки вага повинна змінюватися відносно. Якщо швидкість навчання занадто мала, мережа буде навчатися дуже повільно, а якщо занадто велика, може виникнути коливання, яке не досягне мінімальної точки (рис. 1.2), змінюючи ваги, але не досягнувши оптимального стану. Зазвичай коефіцієнт навчання дуже малий і рухається навколо 0,01. Зміни в алгоритмі зворотного поширення можуть дозволити зменшити величезний коефіцієнт навчання під час навчання. Це призводить до багатьох переваг. Мережа спочатку швидко навчається, оскільки припускається, що вона ініціалізована в далекому стані від ідеального. З часом швидкість навчання знижується, наближаючись до ідеальної мінімальної точки. Збільшення часу навчання біля оптимальної точки спонукає мережу до знаходження оптимального рішення і зменшує ймовірність перенавчання. Якщо навчання розпочинається близько до ідеальної точки, система може спочатку коливатися, але цей ефект зменшується зі зниженням швидкості навчання.

Слід відзначити, що у рівнянні (2.6) змінна представлена вхідним значенням вузла *k* і збігається з вихідним значенням вузла *j*. x_k.

Для покращення процесу оновлення ваги, рівняння (2.6) модифікується таким чином:

$$\Delta w_{j,k}^{n} = l_r \delta_k x_k + \Delta w_{j,k}^{(n-1)} \mu \tag{2,8}$$

У рівнянні 2.8 оновлення ваги на n-ій ітерації визначається за допомогою значення імпульсу, яке множиться на (n-1) ітераційні зміни ваги між вузлами $\Delta w_{j,k}$.

Використання значення імпульсу сприяє прискоренню процесу навчання, стимулюючи вагу змінюватися в тому самому напрямку з великими приростами. Крім того, значення моменту запобігає процесу навчання застрягти в локальному мінімумі, виходячи з нього завдяки автоматичному регулюванню ваги. Зазвичай значення імпульсу лежить в діапазоні від 0 до 1.

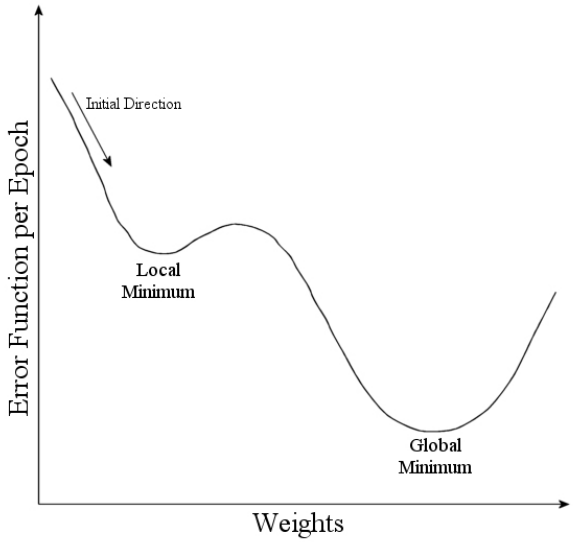

Рисунок 2.1– Глобальні та локальні мінімуми функції розрахунку помилок

У прихованому шарі існують свої особливості в математичних операціях.

Помилку сигналу для вузла j у прихованому шарі можна визначити за наступною формулою:

$$\delta_j = (t_k - O_k) O_k \sum (w_{j,k} \delta_k) \tag{2,9}$$

Сума обчислює взважений сигнал помилки для всіх вузлів k у вихідному шарі. Щодо вихідного шару, формула коригування ваги між вхідним вузлом і та вузлом j має наступний вигляд: $w_{i,j}$

$$\Delta w_{i,j}^{n} = l_r \delta_j x_j + \Delta w_{i,j}^{(n-1)} \mu \tag{2.10}$$

$$w_{i,j} = w_{i,j} + \Delta w_{i,j} \tag{2.11}$$

Підбиваючи підсумок вищесказаного, можна прийти до неприємного висновку, що зворотнє поширення помилки стає важким завданням, якщо припускати, що метою є мінімізація помилок на виході для всіх шаблонів, представлених у нейронній мережі. Наступне рівняння 2.12 використовується для обчислення функції помилок E для всіх стандартів:

$$E = \frac{1}{2} \sum \left(\sum (t_k - O_k)^2 \right) \tag{2.12}$$

У найкращому випадку функція помилок повинна дорівнювати *0*, якщо нейронна мережа коректно навчена. Проте на практиці досягти такого значення є неможливим. Цей алгоритм може бути виражений у формі псевдокоду:

Визначення всіх вхідних та вихідних вузлів Ініціалізація ваг кожного вузла малими випадковими значеннями, зазвичай від -1 до 1. Для кожного шаблону в навчальному наборі виконати наступне:

Подання шаблону в мережі

//Пряме поширення: обчислення вагованої суми для кожного вузла на кожному рівні мережі, додавання порогового значення та розрахунок активації вузлів

кінець

//Зворотне поширення помилки: розрахунок сигналу помилки для вузлів вихідного шару та оновлення ваг кожного вузла в мережі

кінець

//Розрахунок функції помилок Повторювати, поки не досягнута максимальна кількість ітерацій або значення функції помилок не відповідає вказаному порогу

кінець

while ((максимальна кількість ітерацій < задано) I (Функція помилки > вказана))

2.5. Створення моделі

Для створення та аналізу моделі ідентифікації подій в інформаційній безпеці необхідно провести передню підготовку навчальних, контрольних та тестових наборів даних. Загалом модель обробки даних включає чотири основні етапи, які представлені на рис. 2.1:

1. Збір даних про мережеву активність, системні журнали та журнали подій.
2. Обробка вхідних даних та їхнє перетворення в логічну форму, необхідну для подальшого аналізу.
3. Навчання та тестування нейронних мереж.
4. Аналіз отриманих результатів.

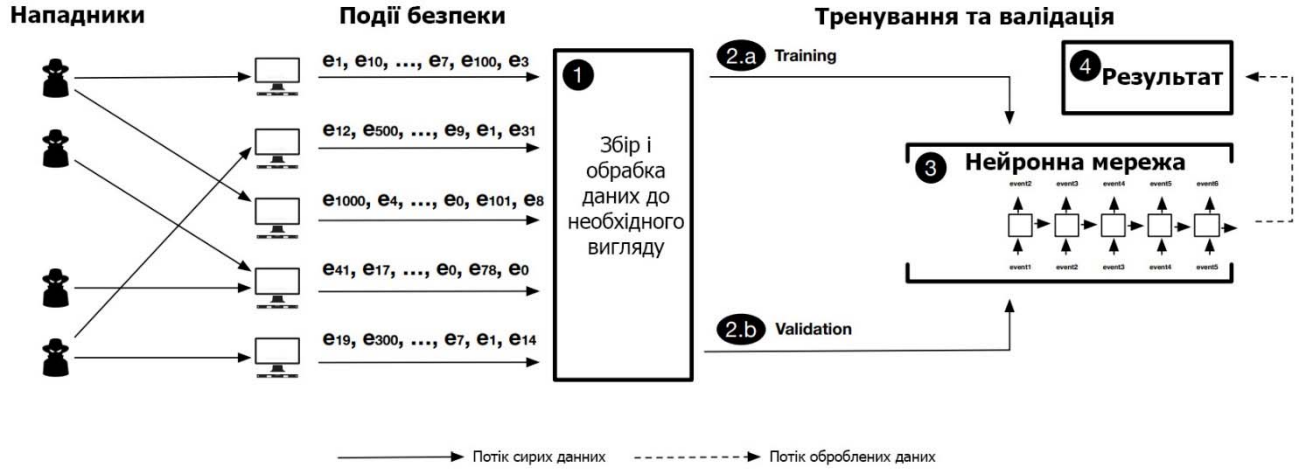

Рисунок 2.1- Схема процесу обробки даних

Ці кроки надають загальний підхід до розробки моделі ідентифікації атак, яка буде використовуватись в подальшому. Сформовані моделі володіють власним уявленням про кожен із етапів і можуть суттєво відрізнятися для виявлення різноманітних типів атак. У цьому розділі детально розглядається модель виявлення різних видів атак у мережі за допомогою аналізу поведінки мережевого трафіку. На

основі отриманих результатів буде розглянуто можливість удосконалення моделей для виявлення конкретних атак.

2.6. Підготовка даних та сценарії атак

Для навчання та тестування інтелектуальної моделі необхідно мати обширний обсяг даних. Оптимальним методом збору даних щодо подій безпеки є використання тієї ж самої цільової системи або мережі, що й об'єкт атаки, разом з мережею зловмисника. Результатом цього процесу є створення спеціалізованого набору даних, який включає інформацію про мережевий трафік, журнали подій, системні журнали і т. д.

У даній статті для моделювання використовувався набір даних Канадського інституту кібербезпеки, який включає сценарії 7 атак: перебір, Heartbleed, ботнет, DoS, DDoS, веб-атаки, проникнення в мережу зсередини. Мережева інфраструктура зловмисника складається з 50 машин, а організація жертви включає 5 відділів, 420 користувальницьких робочих станцій і 30 серверів. Набір даних містить інформацію про перехоплений мережевий трафік і системні журнали для кожної з машин.

Об'єм набору даних становить приблизно 600 ГБ аналітичної інформації, готової до подальшої обробки. Під час аналізу цього набору даних використовуватимуться концепції профілів, а саме: профіль Б та профіль М.

Профіль Б (доброякісний) представляє собою узагальнення поведінки користувача за допомогою різних методів машинного навчання та статистичного аналізу, таких як K-Means, Random Forest, SVM та J48. В інкапсульованих функціях враховуються розмір пакетів протоколу, кількість пакетів на потік, конкретні структури корисного навантаження, розмір корисного навантаження та запит на виділення часу протоколу. В процесі моделювання враховувалися такі протоколи: HTTPS, HTTP, SMTP, POP3, IMAP, SSH та FTP, і, за нашими початковими спостереженнями, більшість трафіку припадає на HTTP і HTTPS.

Профіль М (Злоякісний) описує спробу чітко визначити сценарій атаки. У простій формі люди можуть інтерпретувати ці профілі та виконувати їх. За ідеальних

умов для інтерпретації та виконання цих сценаріїв можуть використовуватися автономні агенти разом із компіляторами. Розглянуто шість різних сценаріїв атак.

1. **Проникнення в мережу зсередини:** У цьому сценарії жертві надіслано шкідливий файл електронною поштою. При успішній експлуатації на комп'ютері жертви запускається бекдор, що призводить до сканування внутрішньої мережі та пошуку нових поштових скриньок.

2. **HTTP-відмова в обслуговуванні:** У цьому сценарії для основної атаки використовувалися інструменти Slowloris і LOIC, які зроблюють веб-сервери недоступними за допомогою однієї атакуючої машини. Атака Slowloris включає повне встановлення TCP-з'єднання з віддаленим сервером та утримання його відкритим, відправляючи регулярні HTTP-запити для уникнення закриття сокетів.

3. **Атакуючі веб-програми:** У цьому сценарії використовувався Damn Vulnerable Web App (DVWA) для перевірки навичок аналізу вразливостей. Спочатку сканувався веб-сайт за допомогою сканера вразливостей веб-застосунків, а потім виконувалися різні типи веб-атак, такі як впровадження SQL, впровадження системних команд та необмежене завантаження файлів.

4. **Атака грубої сили (Атака методом перебору):** Ця атака спрямована на вибір комбінацій імен та паролів для отримання доступу до облікового запису користувача. Для цього існують інструменти, такі як модулі Hydra, Medusa, Ncrack, Metasploit та Nmap NSE, а також програми для злому хешованих паролів, наприклад, hashcat та hashpump.

5. **Атаки останнього оновлення:** Ці атаки ґрунтуються на кількох відомих уразливостях та можуть виконуватися протягом певного часу. Однією з найвідоміших є атака Heartbleed, а інструментом для її проведення є Heartleech. Вона виявляє уразливості та використовує їх для сканування систем.

У таблиці 2.1 представлена інформація про розслідувані атаки.

Таблиця 2.1– Атаки здійсненодля цільових систем

Атака	**Використовувані засоби**
Атака грубої сили	FTP - Патадор SSH – Патадор
DOS атаки	Халк, Золоте Око, Слоулоріс, Повільнийhttptest,п'явка
Веб-атака	Страшенно вразливий веб-додаток (DVWA) Selenium Framework (XSS та брутфорс)
Інфільтраційна атака	Nmap та сканування портів
Атака ботнету	Ares (на базі Python): віддалена оболонка, завантаження/завантаження файлів, захоплення знімків екрану та реєстрація ключів.
DDoS+Сканування портів	Low Orbit Ion Canon (LOIC) для запитів UDP, TCP чи HTTP

2.7. Підготовка даних для навчання нейронної мережі

Є два аспекти аналізу мережевих атак:

Аналіз мережної активності: Вивчення мережевої активності охоплює два основних напрямки: аналіз мережевої активності та аналіз вмісту пакетів даних. Однак аналіз вмісту пакетів даних є складнішим процесом, який буде розглянутий у подальших дослідженнях.

Використання програмного забезпечення CICFlowMeter для аналізу мережної активності: Для аналізу мережної активності використовується програмне забезпечення CICFlowMeter. Це програмне забезпечення, написане на Java, виступає як генератор потоків мережевого трафіку та пропонує гнучку систему управління аналізом трафіку. Програма генерує двонаправлені потоки з вимірюванням 83 статистичних атрибутів, таких як тривалість, кількість пакетів, кількість байтів, довжина пакетів тощо, виміряних як в прямому, так і в зворотному напрямку.

Вихідні дані у форматі CSV містять шість стандартних атрибутів: FlowID, SourceIP, DestinationIP, SourcePort, DestinationPort, Protocol, а також більше 80 статистичних атрибутів за необхідності. Робота програми полягає в генерації цих

атрибутів для кожного потоку, де потоки TCP завершуються після закриття з'єднання, а потоки UDP - після закінчення часу очікування потоку. Тайм-аут потоку може бути призначено індивідуально згідно із визначеною схемою, наприклад, 600 секунд для TCP і UDP.

Після обробки програмним забезпеченням дані містять вибрані та перейменовані для моделювання атрибути, які наведено в таблиці 2.2.

Таблиця 2.2- Атрибути потокумережевий трафік для аналізу

Стандартний порт	порт призначення
Протокол	Протокол
Тимчасова мітка	Час виконання потоку
Тривалість потоку	Тривалість потоку
Усі пакети вперед	Загальна кількість пакетів у напрямку пункту призначення
Усі пакети BWD	Загальна кількість пакетів у зворотному напрямку
Пакети TotLen Fwd	Загальний розмір пакета у напрямку пункту призначення
Пакети TotLen Bwd	Загальний розмір пакета у зворотному напрямку
Передній пкт Лен Макс.	Максимальний розмір пакета у напрямку призначення
Передовий ПКТ Лен Мін	Мінімальний розмір пакета у напрямку призначення
Продовжити Пкт Льон Середній	Середній розмір пакета в цільовому напрямку
Передній пкт Лен Стд	Стандартне відхилення розміру пакета в цільовому напрямку
Бвд Пкт Лен Макс	Максимальний розмір пакета у зворотному напрямку
Bwd Pkt Лен Мін	Мінімальний розмір пакета у зворотному напрямку
Bwd Pkt Len Середній	Середній розмір пакета у зворотному напрямку
Бвд Пкт Лен Стд	Стандартне відхилення розміру пакета у зворотному напрямку
Середній потік IAT	Середній час між двома потоками
Стандартний потік IAT	Стандартне відхилення часу двох потоків
Максимальний потік IAT	Максимальний час між двома потоками

Продовження таблиці 2.2

Мінімальний потік IAT	Мінімальний час між двома потоками
Загальна сума переднього IAT	Загальний час між відправкою двох пакетів до пункту призначення
Середній аванс IAT	Середній час між двома пакетами, надісланими у напрямку пункту призначення.
вперед IAT стандарт	Стандартне відхилення між двома пакетами, відправленими в напрямку пункту призначення.
Передній IAT макс.	Максимальний час між двома пакетами, надісланими у напрямку пункту призначення.
Вперед IAT хв.	Мінімальний час між відправкою двох пакетів до пункту призначення
BWD IAT ТОЙ	Загальний час між двома пакетами, надісланими у зворотному напрямку
Середній IAT	Середній час між двома пакетами, відправленими у зворотному напрямку
BWD IAT стандартний	Стандартне відхилення між двома пакетами, відправленими у зворотному напрямку.
Bwd IAT макс.	Максимальний час між двома пакетами, відправленими у зворотному напрямку
BWD IAT хв.	Мінімальний час між двома пакетами, надісланими у зворотному напрямку
Розширені прапори PSH	Скільки разів прапор PSH був встановлений для пакетів, що прямують до місця призначення (0 для UDP)
Прапори BWD PSH	Скільки разів прапор PSH був встановлений для пакетів, що йдуть у зворотному напрямку (0 для UDP)
URG сигналізує вперед	Скільки разів прапор URG був встановлений для пакетів, що прямують до місця призначення (0 для UDP)
Прапори BWD URG	Скільки разів прапор URG був встановлений для пакетів, що йдуть у зворотному напрямку (0 для UDP)
Передній заголовок len	Загальна кількість байтів, які використовуються для заголовків призначення
Заголовок BWD Льон	Загальна кількість байтів, які використовуються для заголовків у зворотному напрямку.
Пакет «Льон Мін»	Мінімальна довжина потоку

Продовження таблиці 2.2

Льон Макс Пакет	Максимальна довжина потоку
Пкт Льон Середній	Середня довжина потоку
Пкт Лен Стд	Стандартне відхилення довжини потоку
Підкреслити FIN Cnt	Кількість пакетів із FIN
Підкреслити SYN cnt	Кількість пакетів із SYN
Прапор RST Cnt	Кількість пакетів RST
Підкреслити PSH Cnt	Кількість PSH-пакетів
Лічильник сигналізації ACK	Кількість пакетів з ACK
Підкреслити URG Cnt	Кількість пакетів URG
Кількість прапорів CWE	Кількість пакетів CWE
Підкреслити ЄЕК Cnt	Кількість упаковок з ЄЕК
Співвідношення зниження/підвищення	Коефіцієнт завантаження/розвантаження
Середній розмір упаковки	Середній розмір упаковки
Середній розмір сегмента фіда	Середній розмір пакета в цільовому напрямку
Середній розмір Bwd Mon	Середній розмір пакета у зворотному напрямку
Пакети пересилання підтоків	Середня кількість пакетів у підпотуці у напрямку пункту призначення
Байти пересилання підпотоку	Середня кількість байтів у підпотуці у напрямку призначення
Пакети Underflow BWD	Середня кількість пакетів у підпотоці у зворотному напрямку
Підтік BWD байтів	Середня кількість байт у підпотоці у зворотному напрямку
Завантаження вперед Win Byts	Кількість байтів, відправлених у початковому вікні у прямому напрямку
Ініціалізація байтів Bwd Win	Кількість байт, відправлених у початковому вікні у зворотному напрямку
Пакети даних FWD Act	Кількість пакетів з мінімум 1 байтом корисних даних TCP у напрямі призначення.

Продовження таблиці 2.2

Активний середній	Середній час, протягом якого потік був активний, перш ніж він став вільним
Активний шаблон	Стандартне відхилення часу, протягом якого потік був активним, перш ніж він став безкоштовним.
Максимальний актив	Максимальний час, протягом якого потік був активний, перш ніж він став бездіяльним
Мінімальна активність	Мінімальний час, протягом якого потік був активний, перш ніж він став бездіяльним.
Середній холостий хід	Середній час простою потоку перед тим, як стати активним
Шаблон простою	Стандартне відхилення часу, протягом якого потік був неактивним, як стати активним.
Максимальний холостий хід	Максимальний час, протягом якого потік простоював, як стати активним.
Мінімальний холостий хід	Мінімальний час, протягом якого потік простоював, як стати активним.

Після створення датасету кожен потік був відзначений відповідно до того, чи сталася атака в момент потоку, або як позитивний, якщо атака не відбулася (звичайне використання сервісів). У кінцевий датасет був доданий новий атрибут "Label", що вказує, чи потік відповідає конкретній атаці або позитивному трафіку. Атрибут "Timestamp" було виключено, оскільки він не несе інформаційної цінності. Дані були позначені відповідно до таких значень: Доброякісний, FTP-BruteForce, SSH-брутфорс, DoS-GoldenEye, DoS-Slowloris, Тест DoS-SlowHTTP, DoS-Халк, DDoS-LOIC-HTTP-атаки, DDoS-LOIC-UDP, DDoS-HOIC, Brute Force-Web, Brute Force-XSS, SQL-ін'єкція, проникнення, бот.

2.8. Підготовка моделі до навчання

Синтез нейромережевої моделі було виконано з урахуванням багатошарового перцептрону Румельхарта.

Багатошаровий перцептрон Румельхарта – це варіант перцептрону Розенблатта, в якому алгоритм зворотного розповсюдження помилки застосовується до навчання

всіх шарів. На жаль, назва не відображає специфіку цього типу перцептрону, оскільки вона не пов'язана із кількістю шарів (зазначимо, що перцептрон Розенблатта також мав кілька шарів). Потреба в багатьох шарах у навчанні виникає тому, що теоретично одного прихованого шару достатньо для кодування вхідного сигналу та отримання лінійної картини вихідного сигналу. Але використання більшої кількості шарів може дозволити зменшити кількість елементів у кожному, що призводить до покращення якості навчання.

Даний алгоритм реалізовано у програмному забезпеченні Weka.

Weka — це програмне забезпечення з відкритим вихідним кодом, розповсюджене під ліцензією GNU General Public License, яке включає набір алгоритмів машинного навчання для інтелектуального аналізу даних. Серед його функцій – інструменти для підготовки, класифікації, регресії, кластеризації даних, аналізу правил асоціацій та візуалізації.

Weka має API, написаний на Java, яке реалізує існуючі алгоритми навчання моделей нейронних мереж. Залишається завданням описати та сформувати процес моделювання та підготувати дані для навчання. Було створено спеціальне програмне забезпечення для опису процесів навчання та тестування цієї моделі, а схематичну логіку можна представити, як показано в таблиці 2.3.

Таблиця 2.2– Опис алгоритму навчання нейронної мережі

Ініціалізація моделі класу
`Генератор моделеймг"="новийГенератор моделей();`
ModelGenerator є основним класом генератора моделей. У нього включені функції завантаження даних в оперативну пам'ять для подальшої обробки, ініціалізації класу нейронної мережі, проведення тестування та збереження моделі для майбутнього використання.
Завантаження даних для навчання в оперативну пам'ять
`Примірникинабір даних"="мг.завантажити набір даних(ШЛЯХ НАБОРУ ДАНИХ);`
Функція **loadDataset** входить до складу класу **ModelGenerator** і втілює інтерфейс завантаження даних із файлу, який попередньо підготовлено у форматі .arff. Також вона ініціалізує клас Instances, який описаний у Weka API.

Продовження таблиці 2.3

Визначення номіналів за числовими значеннями
ФільтрnumToNominalFilter"="новийчисловий-номінальний(); Нитка[]параметри"="новийНитка[два]; ... numToNominalFilter.ВстановитиОпції(Параметри); numToNominalFilter.setInputFormat(набір даних); набір даних =Фільтр.використовуватиФільтр(набір даних, numToNominalFilter);
Нейронна мережа репрезентує числові значення заданої залежності, відповідно налаштовуючи ваги для відповідних величин. У процесі нормалізації даних взаємозв'язки між конкретними атрибутами можуть бути неправильно інтерпретовані, що може ускладнити процес навчання. Для критичних числових атрибутів, таких як значення портів (від 1 до 65565), необхідно використовувати інший метод представлення. Для вирішення цієї проблеми використовується конвертація числових значень у номінальні.
Нормалізація даних
Фільтрфільтр"="новийНормалізувати(); фільтр.setInputFormat(набір даних); Примірникинабір даних нор"="Фільтр.використовуватиФільтр(Набір даних, фільтр);
Нормалізація даних в нейронних мережах — це процес оптимізації значень у наборі даних числового типу, перетворюючи їх із широкого діапазону в діапазон від 0 до 1, зберігаючи пропорційність. Нормовані значення суттєво підвищують швидкість навчання моделі, але можуть вплинути на точність навчання.
Поширення навчальних та тестових зразків
внутрішнійрозмір поїзда"="внутрішній)Математика.круглий(набір даних.кількість екземплярів()*0,9); внутрішнійtestSize"="набір даних.кількість екземплярів() - Розмір поїзда; Примірникинабір даних поїзда"="новийПримірники(ні набір даних,0, Розмір поїзда); Примірникитестовий набір даних"="новийПримірники(набір даних, trainSize, testSize);
Розподіл вибірки є необхідним для перевірки початкової адекватності моделі. У цьому дослідженні навчальна вибірка становила 90% від загального обсягу переданих для навчання даних..
Ініціалізація процесу навчання нейронної мережі
Багатошаровий персептронA-H-A= (Багатошаровий персептрон)мг.buildClassifier(Набір навчальних даних);

Продовження таблиці 2.3

Багатошаровий персептрон - це API-клас у Weka, який визначає математичну структуру нейронної мережі, базуючись на концепції багатошарового персептрону. Функція buildClassifier, описана в генераторі моделей, ініціалізує екземпляр класу нейронної мережі з необхідними параметрами і виконує його навчання.
Початкове тестування моделі
`Ниткарезюме оцінки"="мг.ставкаМодель(Енн, набір даних поїзда, набір тестових даних);`
Функція "ставка Модель" використовує тестовий набір для класифікації даних і порівнює отримані значення з моделлю. Результати успішних порівнянь використовуються для формування значень помилок навчання.

Детальні програмні коди наведено у додатку А

2.9. Модель навчання нейронної мережі

Для досягнення необхідної точності ідентифікації подій, відносна похибка повинна перевищувати 5%.

З урахуванням великого обсягу навчальних даних, їх було розділено на блоки за типами атак, і було вирішено створити окрему нейромережеву модель для ідентифікації кожного типу атаки.

На початковому етапі дослідження було здійснено синтез моделі ідентифікації атаки методом перебору FTP-SSH на основі спеціалізованого набору даних, інформація про який наведена на рис. 2.2.

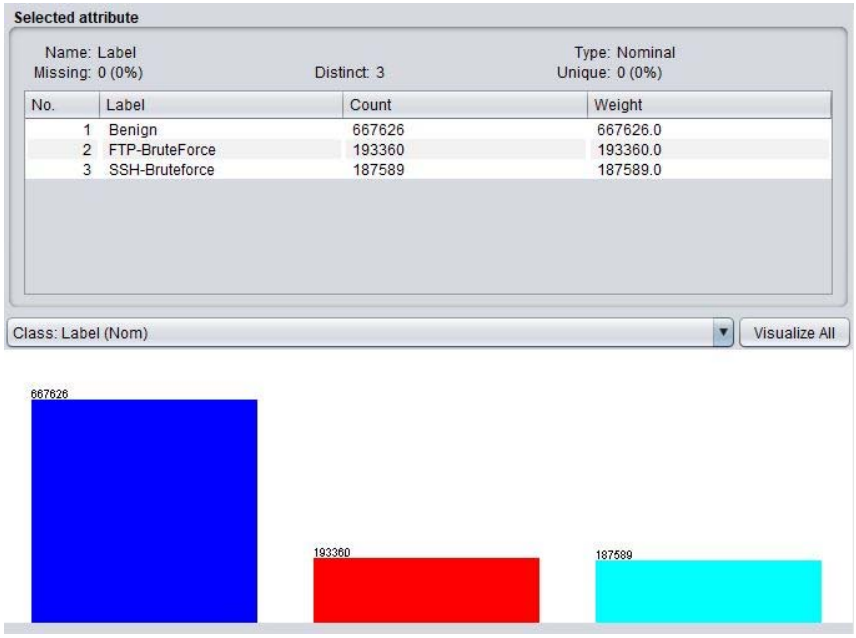

Рисунок 2.2– Набір даних для навчання моделі виявлення атак грубої сили FTP-SSH.

Приклад на рис.2.3. показує ініціалізацію нейронного алгоритму з параметрами:

```
public Classifier buildClassifier(Instances traindataset) {
    MultilayerPerceptron m = new MultilayerPerceptron();

    // m.setGUI(true);
    // m.setValidationSetSize(0);
    // m.setBatchSize("100");
    // m.setLearningRate(0.3);
    // m.setSeed(0);
    // m.setMomentum(0.2);
    m.setTrainingTime(50);// тривалість навчання (кількість епох)
    // m.setNormalizeAttributes(true);
  // m.setHiddenLayers("2,3,3") три приховані шари з 2 вузлами в першому шарі, 3 вузла в другому і 3 вузли

    /*...
    try {
        m.buildClassifier(traindataset);

    } catch (Exception ex) {
        Logger.getLogger(ModelGenerator.class.getName()).log(Level.SEVERE, null, ex);
    }
    return m;
}
```

Рисунок 2.3– Подання ініціалізації алгоритму та його параметрів

Швидкість навчання за замовчуванням має бути встановлена у діапазоні від 0 до 1, ініційована значенням за замовчуванням = 0,2.

Частота пульсу повинна бути встановлена у діапазоні від 0 до 1, зі значенням за замовчуванням = 0,3.

Кількість епох для навчання за замовчуванням становить 500-600, проте використовується 50-100 через обсяг даних.

Відсотковий розмір набору перевірки, що використовується для припинення навчання, повинен бути у діапазоні від 0 до 100, за замовчуванням = 0.

Значення для генерації генератора випадкових чисел повинно бути більше 0 і менше довжини.

Кількість прихованих шарів має бути списком натуральних чисел або букв, розділених комами. 'a' = (атрибути + класи) / 2, 'i' = атрибути, 'o' = класи, 't' = атрибути + класи. За замовчуванням – «a».

Розмір стандартного пакета, необхідного для прогнозування, за замовчуванням = 100.

Для оцінки точності моделі вихідний набір даних був розділений на менші групи: 100 000, 250 000, 500 000, 1 000 000 відповідно. Результати навчання моделі на наборі даних із 100 000 записів представлено на рис. 2.4.

```
Evaluation: Summary
Correctly Classified Instances       19893               99.465  %
Incorrectly Classified Instances       107                0.535  %
Kappa statistic                          0.9889
K&B Relative Info Score            1927505.1563 %
K&B Information Score                23721.0376 bits      1.1861 bits/instance
Class complexity | order 0           24377.6345 bits      1.2189 bits/instance
Class complexity | scheme             1035.0185 bits      0.0518 bits/instance
Complexity improvement     (Sf)      23342.616  bits      1.1671 bits/instance
Mean absolute error                      0.0118
Root mean squared error                  0.0606
Relative absolute error                  3.6326 %
Root relative squared error             15.0765 %
Total Number of Instances            20000

Detailed Accuracy By Class
                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1,000    0,003    0,999      1,000    0,999      0,998    0,998     0,996     SSH-Bruteforce
                 0,971    0,000    1,000      0,971    0,985      0,982    0,994     0,987     Benign
                 1,000    0,005    0,969      1,000    0,984      0,982    0,996     0,957     FTP-BruteForce
Weighted Avg.    0,995    0,002    0,995      0,995    0,995      0,993    0,997     0,989

Confusion Matrix
     a     b     c   <-- classified as
 13625     0     5 |     a = SSH-Bruteforce
    16  3425    86 |     b = Benign
     0     0  2843 |     c = FTP-BruteForce
```

Рисуно 2.4– Результат навчання моделі та її перевірки на 20% вихідного набору даних.

Програма створила файл із розширенням .bin, і середній час на створення моделі для цього набору даних склав 400 секунд. Цей файл буде використовуватися для класифікації нових даних.

Був розроблений спеціальний метод для класифікації атак з використанням синтезованої моделі ClassifyInstance, і алгоритм його роботи представлений у таблиці 2.4.

Таблиця 2.3– Опис алгоритму за допомогою нейронної мережі

Ініціалізація моделі класу
`Генератор моделеймг"="новийГенератор моделей();`
Ініціалізувати новий екземпляр класу
Завантаження даних для навчання в оперативну пам'ять
`Примірникинабір даних"="мг.завантажити набір даних(ПЕРЕВІРКА ШЛЯХУ);`
Результати функції приведені в таблиці 2.4.
Визначення номіналів за числовими значеннями

`ФільтрnumToNominalFilter"="новийчисловий-номінальний();`
`Нитка[]параметри"="новийНитка[два];`
`...`
`numToNominalFilter.ВстановитиОпції(Параметри);`
`numToNominalFilter.setInputFormat(набір даних);`
`набір даних =Фільтр.використовуватиФільтр(набір даних, numToNominalFilter);`

Результати функції приведені в таблиці 2.4
Нормалізація даних

`Фільтрфільтр"="новийНормалізувати();`
`фільтр.setInputFormat(набір даних);`
`Примірникинабір даних нор"="Фільтр.використовуватиФільтр(Набір даних, фільтр);`

Результати функції приведені в таблиці 2.4.
Ініціалізація класифікатора
`МодельКласифікаторклс"="новийМодельКласифікатор();`
Розроблено клас "МодельКласифікатор", який реалізує інтерфейс відповідності значення виведення нейронної мережі визначеному типу атаки.
Ініціалізація валідатора
`Валідатор[]валідатори"="новийВалідатор[] {новийВалідатор(«Доброякісний»),новий Валідатор("ФТП-BruteForce"),новийВалідатор("СШ-Брутфорс")};`

Продовження таблиці 2.4

Валідатори визначають категорії мережі та використовуються для відображення користувачу значення рейтингу.
Класифікація моделей
`Ниткаім'я класу"="клс.ранжувати(набір даних.брати(i), ШЛЯХ МОДЕЛІ);`
Отримання відповіді від нейромережі
Визначення помилок мережі
`Ниткаклас за замовчуванням"="набір даних.брати(Я).рядкове значення(66);` `Валідатор.оновленняВалідація(Валідатори, defaultClass, ім'я класу);`
Якщо дані, що надходять для оцінки класифікатора, включають збігаючіся шаблони, можна розрахувати відносну помилку мережі.

Для ілюстрації роботи новоствореної моделі було обрано 15 записів, які включають 5 випадків Brute Force FTP, 5 випадків Beign (звичайний трафік) та 5 випадків Brute Force SSH Flows відповідно. За замовчуванням всі дані були спеціально відзначені як FTP-атака методом перебору. Приклад наведено на рисунку 2.5.

```
@data
21,6,1,1,1,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,40,20,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,1,
21,6,2,1,1,0,0,0,0,0,0,0,0,0,0,2,0,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,40,20,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,1,
21,6,1,1,1,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,40,20,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,1,
21,6,2,1,1,0,0,0,0,0,0,0,0,0,0,2,0,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,40,20,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,1,
21,6,1,1,1,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,40,20,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,1,
67,17,685,1,1,300,329,300,300,300,0,329,329,329,0,685,0,685,685,0,0,0,0,0,0,0,0,0,0,0,0,0,8,8,300,329,309
0,0,112640738,3,0,0,0,0,0,0,0,0,0,0,56320369,94.752309,56320436,56320302,112640738,56320369,94.752309,563
0,0,112640743,3,0,0,0,0,0,0,0,0,0,0,56320371.5,13.435029,56320381,56320362,112640743,56320371.5,13.435029
0,0,112640627,3,0,0,0,0,0,0,0,0,0,0,56320313.5,4401.739713,56323426,56317201,112640627,56320313.5,4401.73
0,0,112640733,3,0,0,0,0,0,0,0,0,0,0,56320366.5,51.618795,56320403,56320330,112640733,56320366.5,51.618795
22,6,555650,22,22,1928,2665,640,0,87.636364,137.780552,976,0,121.136364,258.64156,12922.093023,38554.860045
22,6,560585,22,22,1944,2665,640,0,88.363636,137.869001,976,0,121.136364,258.64156,13036.860465,48855.286144
22,6,530547,22,20,1928,2665,640,0,87.636364,137.780552,976,0,133.25,268.771254,12940.170732,45365.27566,271
22,6,533609,22,21,1960,2665,640,0,89.090909,137.953377,976,0,126.904762,263.574639,12704.97619,42668.192189
22,6,565469,22,21,1944,2665,640,0,88.363636,137.869001,976,0,126.904762,263.574639,13463.547619,45697.10508
```

Рисунок 2.5– Дані для тестування моделей та класифікації атак.

Після активації класифікатора виділяються відповідності потоку даних і типу атаки, як це показано на рисунку 2.6.

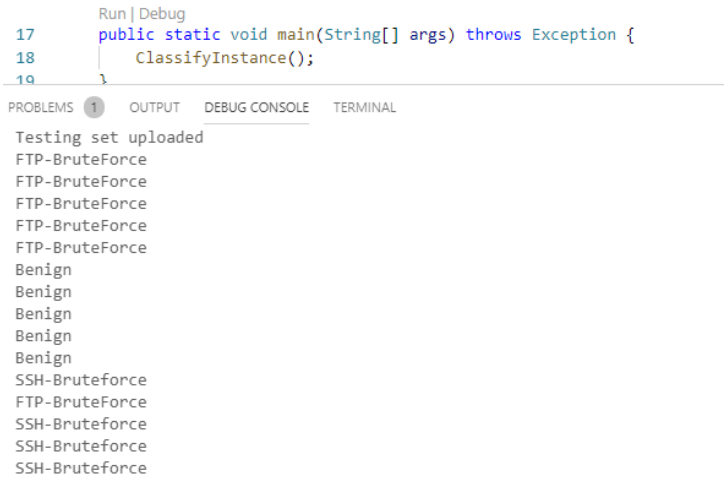

Рисунок 2.6– Приклад класифікації атак

Лише одна помилка виникла під час атаки: FTP Bruteforce було невірно визначено як SSH-Bruteforce. Це демонструє, як працює мережа, але не відображає її реальну життєздатність. Для підтвердження його ефективності його випробували на обширних наборах даних та інших типах атак.

2.10. Адекватності отриманої моделі

Для випробування моделі використовувалися два додаткові набори даних для кожного типу. Після виконання класифікації була отримана наступна інформація щодо SSH- та FTP-Bruteforce-атак для набору, що складається із ста тисяч об'єктів, яка відображена на рисунку 2.7.

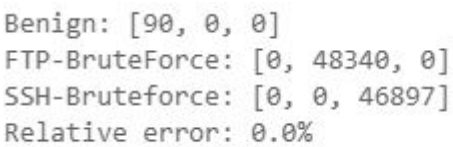

Рисунок 2.7 – Приклад тестування отриманої моделі

Повна інформація про кожен тип атаки та результати навчання надані у таблиці 2.5.

Таблиця 2.4- Результат навчання моделі

Назва атаки	Підготовлений до навчання	Тестовий набір 1	Тестовий набір 2	Відносна погрішність, %
FTP-брутфорс	96670	48330/48330	48200/48339	0,14
SSH-брутфорс	93785	46887/46886	46801/46896	0,2
Дос атакує GoldenEye	20743	199/10387	331/10377	96
Дос атакує Слолориса	5485	2/2648	0/2748	97
Dos атакує LOIC-UDP	765	0/423	0/432	98
Dos атакує HOIC	343008	171403/171403	171503/171503	0
Дві атаки ХАЛКА	230943	114478/114478	115478/115478	0
Доброякісний HTTP	172022	18325/87006	15468/86005	81
Проникнення	45487	0/22836	0/22736	100
Ботне	140821	70507/70509	70405/70408	0,02
Доброякісний Інфобот	468905	234502/236902	234902/234902	0

Створені моделі та докладні вихідні коди, включаючи набори тестових даних для кожної категорії атаки, подані у Додатку Б.

Висновки по другому розділу

У даному розділі дипломної роботи розглянуті основні архітектури нейронних мереж, що сформульовані протягом періоду з кінця 20 століття до початку 21 століття. Кожна з представлених архітектур вирішує концептуальні задачі унікальним способом. Під час виконання роботи було вирішено використовувати архітектуру нейронної мережі, засновану на багатошаровому персептроні "Румельхарта". Основною метою було визначення типу атаки для кожного шаблону вхідного мережевого трафіку.

Для створення моделей даних використовувалось програмне забезпечення, також включено витратомір CICFlowMeter та додаткове програмне забезпечення, розроблене на мові програмування Python. Опубліковані набори даних про мережеві атаки використовувалися як джерело вхідних даних.

Розглянуто концепцію побудови нейронної мережі, а також наведено результати її роботи. З таблиці 2.5 можна зробити висновок, що дана модель ефективніше виявляє брутфорс-атаки на служби FTP та SSH, а також Dos-атаки, такі як HOIC, HULK та активність ботнетів.

3. РОЗРОБКА МОДЕЛІ ІДЕНТИФІКАЦІЇ АТАКИ НА МЕРЕЖІВІ СЕРВІСИ

3.1. Атаки на мережеві сервіси

Веб-сервіс – це сервіс, який функціонує за протоколом HTTP (протокол передачі гіпертексту), обробляє запити на певному мережевому порту, обслуговує веб-документи (HTML, JSON, XML, зображення) і використовується для вирішення конкретних завдань та проблем у світовій мережі.

HTTP є одним із найбільш поширених протоколів в Інтернеті, відноситься до протоколів моделі OSI 7-го рівня додатків. За міжнародними даними, кількість опублікованих в Інтернеті веб-сайтів у 2020 році перевищила 1,7 мільярда [19].

Згідно з методологією OWASP (Open Web Application Security Project), можна виділити 10 найбільш поширених кібератак на веб-сервіси. Перш за все, з точки зору поширеності та критичності, слід зазначити ін'єкцію. Такі види впровадження, як SQL, NoSQL, OS та LDAP, виникають, коли ненадійні дані передаються інтерпретатору як частині команди чи запиту та виконуються, що надає доступ до критичних ресурсів чи служб.

Інші найважливіші кібератаки на веб-сервіси зазначені нижче:

1. **Порушена автентифікація:** Програмні функції, пов'язані з автентифікацією та управлінням сеансами користувачів, часто реалізуються неправильно, що дозволяє зловмисникам отримати доступ до паролів, ключів чи токенів.

2. **Витік конфіденційних даних:** Багато веб-програм та служб API не мають належного захисту інформації, що призводить до розголошення конфіденційної інформації через передачу даних у відкритому форматі або помилки у налаштуванні доступу.

3. **Зовнішні об'єкти XML (XXE):** Недоліки в обробці зовнішніх ресурсів у документах XML можуть призвести до витоку внутрішніх файлів та інших атак.

4. **Порушення режиму контролю доступу:** Неправильна реалізація обмежень, що визначають контроль доступу до ресурсів, часто використовується зловмисниками для отримання несанкціонованого доступу.

5. **Неправильне налаштування безпеки:** Часто це є наслідком використання налаштувань за замовчуванням, недостатніх або неправильних налаштувань користувача, помилкових HTTP-заголовків та відображення помилок.

6. **Міжсайтовий скриптинг XSS:** Атаки XSS стають можливими, коли програма включає ненадійні дані на веб-сторінку без належної перевірки, що дозволяє виконання зловмисницьких сценаріїв у браузері користувача.

7. **Небезпечна десеріалізація даних:** Небезпечна десеріалізація може призводити до віддаленого виконання коду, що робить систему вразливою для атак.

8. **Використання компонентів із відомими вразливостями:** Експлуатація вразливих компонентів може призвести до компрометації сервера чи втрати даних.

9. **Недостатній облік та контроль:** Недостатній моніторинг та відсутність ефективного реагування на ІТ-інциденти дозволяють зловмисникам тривати в атаках.

10. **Атака за допомогою компрометованих компонентів:** Використання компрометованих компонентів у системі може викликати серйозні наслідки для безпеки.

Аналізуючи цей контекст та результати попереднього розділу, було обрано детальне вивчення моделі одного з типів атак на веб-сервіси, а саме SQL-ін'єкцій.

3.2. Розробка моделі ідентифікації SQL-атак

3.2.1. Специфікація атак

SQL-ін'єкція, або атака мовою структурованих запитів, представляє собою стратегію зміни запитів до бази даних через використання вразливостей у веб-додатках. У випадку успішної атаки, зловмиснику відкривається можливість отримати, змінити або навіть видалити конфіденційну інформацію.

Вразливості, пов'язані з використанням SQL-коду, можуть виникнути в будь-якому веб-додатку, який використовує реляційні бази даних, такі як MySQL, MSSQL,

Oracle, SQL Server, PostgreSQL і так далі. Атаки, що використовують SQL-ін'єкції (SQLi), входять до переліку десяти найпоширеніших атак згідно з методологією OWASP.

SQL є мовою запитів, спрямованою на управління даними в реляційних базах даних, що є основою для багатьох веб-сайтів. У деяких випадках SQL може використовуватися для виконання команд операційної системи, тому успішна атака SQL-ін'єкцією може призвести до серйозних наслідків.

Веб-сторінка чи веб-програма, яка є вразливою до атаки SQL-ін'єкцією, використовує введені користувачем параметри безпосередньо в SQL-запитах. Зловмисник може змінювати отримані дані, які часто називають корисним навантаженням, і які є ключовою частиною атаки. Після відправлення зловмисником спеціально створеного SQL-запиту виконуються незаплановані команди [16].

Зазвичай формат SQL-запиту виглядає наступним чином:

vbnetCopy code

SELECT ID, ім'я, прізвище FROM автори;

Цей запит призначений для отримання стовпців «ID», «ім'я» та «прізвище» з таблиці «автори». Результатом відповіді бази даних будуть рядки таблиці. Для конкретизації відповіді сервера вказується запит:

sqlCopy code

SELECT ідентифікатор, ім'я, прізвище FROM автори WHERE ім'я = 'Джон' AND прізвище = 'Сміт';

Важливо відзначити, що рядкові параметри обгортаються одинарними лапками. Враховуючи можливість генерації цих значень на основі користувацького вводу (наприклад, з веб-форми), зловмисник може спробувати змінити структуру вихідного SQL-запиту, вставивши такі параметри: Ім'я: Джейсон ім'я: Смітті Результатовий запит буде виглядати так:

sqlCopy code

SELECT ідентифікатор, ім'я, прізвище FROM автори WHERE ім'я = 'Джейсон' AND прізвище = 'Смітті';

При виконанні цього запиту база даних може повернути помилку, подібну до такої:

makefileCopy code

Сервер: Повідомлення 170, Рівень 15, Стан 1, Рядок 1. Рядок 1: неправильний синтаксис поруч із «hn».

Існує кілька стратегій для виконання атак з використанням SQL-ін'єкцій:

- Внутрішньосмуговий SQLi використовує помилки бази даних або команди UNION для отримання несанкціонованого доступу;
- Сліпий SQLi аналізує відповіді бази даних за допомогою передачі логічних операцій;
- Out-of-band SQLi використовує різні канали для виконання атак та збору інформації, зазвичай, це залежить від функціоналу, активованого на сервері бази даних [21]. Для зменшення впливу цих атак на функціонування бізнес-сервісів і забезпечення безпеки кінцевих користувачів розробники, системні адміністратори та DevOps повинні приймати ряд заходів безпеки. Ці заходи включають у себе фільтрацію та блокування запитів, які можуть містити потенційно шкідливий контент і спричинити SQL-атаку. В роботі розглядається створення модуля HTTP (Hyper Text Transfer Protocol), який може ідентифікувати атаки типу SQL-ін'єкція за допомогою штучних нейронних мереж.

3.2.2. Опис моделі

Основною метою дослідження було створення логічного модуля, спроможного асоціювати вхідний HTTP-запит із SQLi-атакою. Розроблена модель складається з трьох основних компонентів:

1. Генератор URL-адрес (Uniform Resource Locator).
2. Класифікатор URL.
3. Модель на основі нейронної мережі.

Генератор URL-адрес є тестовим модулем, необхідним для формування вихідного набору даних. Він включає два компоненти: генерацію звичайних URL-

адрес за допомогою збору інформації з популярних веб-сайтів, таких як парсинг файлу sitemap.xml, і генерацію шкідливих адрес за допомогою додавання типових SQL-ін'єкційних параметрів до URL-адрес, отриманих попереднім методом. Альтернативний метод для заповнення вибірки - використання наборів даних з відкритим вихідним кодом.

Типові запити SQL-ін'єкцій містять ключові слова, що зазвичай використовуються для виконання операцій з таблицями бази даних SQL, стосуються як всієї бази даних, так і окремих таблиць.

Для синтезу та аналізу моделі ідентифікації атак потрібна попередня підготовка наборів навчальних, контрольних та тестових даних. Навчальний набір містить параметри об'єкта навчання, вибір яких здійснювався евристично на основі аналізу суттєвих сигналів атаки, які можуть містити URL.

Оскільки моделі на основі нейронних мереж працюють лише з числовими даними, представленими в деякому числовому діапазоні, був розроблений класифікатор URL, який перетворює URL в двійковий формат і визначає логічний ідентифікатор "істина (1)", якщо адреса відноситься до атаки, і "брехня (0)" в іншому випадку. Таким чином, для кожного URL формувався вхідний вектор, а параметри для аналізу представлені в таблиці 3.1.

Таблиця 3.1- Номіновано вектори для шаблонів параметрів SQL

Номер параметра вхідного вектора	Параметр запиту
x1	'(одинарні лапки)
x2	СТВОРЮВАТИ
x3	ВИДАЛИТИ З
x4	ТАБЛИЦЯ ФОЛСІВ
x5	ВСТАВИТИ У
x6	ЄДНІСТЬ
x7	І
x8	АБО
x9	-
x10	(лазівка)
x11	У
x12	ВИКОНАВЕЦЬ

Отже, структуру вхідного вектора можна виразити так:

$$y^T = [y_1 y_2 \cdots y_n]$$

Наприклад, якщо у нас є 12 параметрів у URL-адресі, і цей URL містить вираз "CREATE TABLE... AND INSERT INTO...", то його вектор буде виглядати наступним чином: 010010100100.

При такому підході модель нейронної мережі матиме 12 нейронів на вході. Кожен елемент вхідного вектора визначається параметром: "Доброякісний" (0) - якщо не відноситься до атаки, і "Ін'єкція" (1) - якщо відноситься до атаки. Для представлення цієї інформації на виході достатньо мати лише один нейрон, який розділить вхідні вектори на два класи: 0 та 1.

3.2.3. Навчання та аналіз отриманих результатів

Для розробки програмного забезпечення з використанням мови програмування Node.js для навчання моделей використовувався внутрішній модуль Synaptic. Synaptic - це бібліотека нейронних мереж JavaScript для node.js, призначена для створення та навчання різних типів нейронних мереж.

Отримана модель має три шари нейронів. Нейрони вхідного та прихованого шарів використовують сигмоподібну функцію активації, тоді як вихідний шар використовує лінійну функцію. Архітектура моделі включає вхідний шар з 12 нейронами, прихований шар з 6 нейронами, та вихідний шар з сигналом, що дорівнює 1. Вхідний вектор, що складається з 12 елементів, обробляється моделлю з використанням меж припустимих значень [0, 1]. Ініціалізація моделі проводилася випадковими значеннями з інтервалу [-1, 1].

Кількість нейронів у прихованому шарі визначалася двома правилами: "Ідеальний розмір прихованого шару" - це значення між кількістю вузлів у вхідному та вихідному шарі, і "Кількість нейронів у цьому шарі" - середня кількість нейронів у вихідному та вхідному шарах.

Для налаштування нейронної мережі було визначено кілька параметрів, таких як швидкість навчання (від 0 до 1), максимальна кількість ітерацій (не більше 200), та мінімальна помилка (0,005).

Вихідна вибірка була розділена на навчальну (70%), контрольну (15%) та тестову (15%) частини. Згенеровано 30132 URL-адреси з різноманітністю. Навчальний набір містив 20172 записи, контрольний - 5035, та тестовий - 5028. З навчанням синтезована модель досягла точності класифікації близько 95%. Графік навчання демонструє зміну помилки з кожною ітерацією, яка проводилася на різних розмірах вибірок.

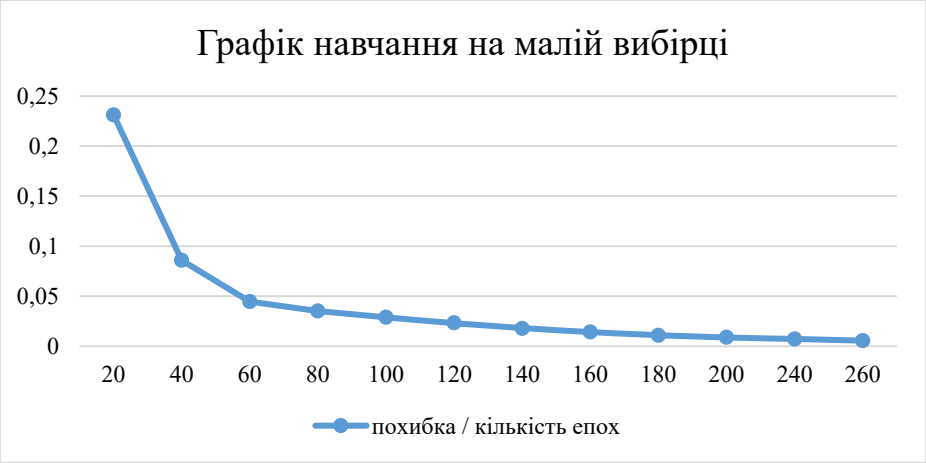

Рисунок 3.1– Графік навчання у малий вибірці

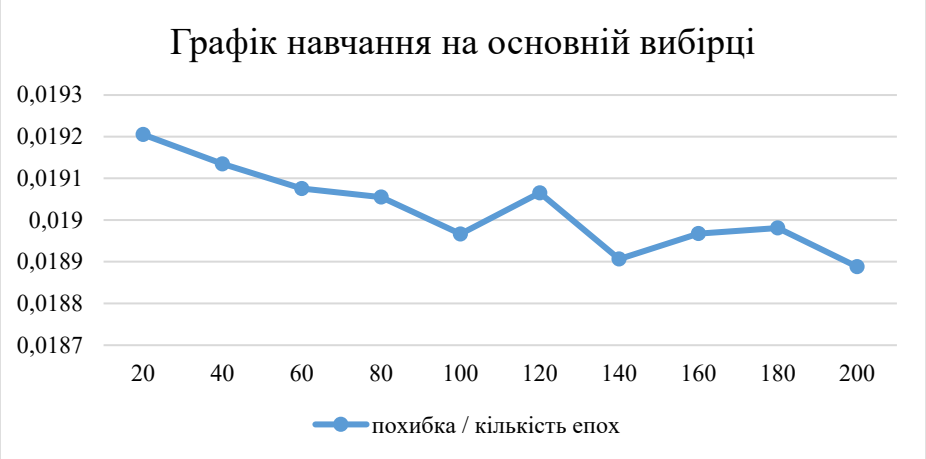

Рисунок 3.2- Графік дослідження основної вибірки

На зазначеному графіку видно, що відносна помилка навчання для основної навчальної вибірки становить 1,9%.

Під час функціонування нейронної мережі може виникнути ситуація, коли параметри SQL-ін'єкцій, які аналізуються, вказані в URL-адресі в довільному порядку і не завдають ніякої шкоди веб-серверу та базі даних відповідно (False Positive), але система визначає їх як атаку. У подальших дослідженнях для вирішення цієї проблеми може бути використано явні шаблони впровадження SQL і враховано HTTP-коди стану відповідей веб-сервера.

Отже, підсумовуючи вищезазначене, відносна помилка синтезованої моделі на контрольній та тестовій вибірках не перевищує 5%. Результати класифікації представлені в таблиці 3.3.

Таблиця 3.3 – Результат навчання моделі нашої моделі

Тип трафіку	Освітній	Контроль (успішний/повний)	Тест (успішний/завершений)	Відносна погрішність, %
Звичний	12923	3186/3227	3180/3117	0,8%
SQLi	7270	1720/1807 р.	1731/1709 р.	4,9%

Висновки по третьому розділі

Отже, проведений аналіз результатів моделювання на тестовому наборі даних дозволяє зробити висновок, що у цілому запропонований метод виявлення SQL-ін'єкцій забезпечує можливість виявлення атак даного типу з точністю 95%. За нашим переконанням, подальше підвищення точності ідентифікації можливе за умови навчання моделі на значно більших обсягах даних.

4. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

4.1. Екологічне законодавство

Протягом тисячоліть вплив людства на навколишнє середовище був невеликим, але у другій половині XX століття зростаюче антропогенне навантаження призвело до серйозних екологічних проблем. Підвищена увага до проблеми забезпечення економічних та соціальних потреб, а також збереження довкілля призвела до прийняття законодавчих актів у багатьох країнах. Зараз розробляються та впроваджуються нові технології для мінімізації шкідливого впливу виробничих процесів на повітря, воду та ґрунт.

Охорона природи є нашим обов'язком, оскільки природа надає нам багатства, і ми повинні витрачати їх розумно, зберігаючи баланс. Господарська діяльність людини часто завдає шкоди навколишньому середовищу: вирубуються ліси для будівництва та експансії сільськогосподарських угідь, щоб виробляти папір та меблі.

Транспорт та промисловість спричиняють забруднення повітря, води та ґрунту. Практично в усіх країнах світу існують закони про охорону природи, які зобов'язують людей берегти природні ресурси та забезпечувати екологічну безпеку.

Процеси глобалізації та суспільних трансформацій підняли проблему збереження довкілля на перший план, вимагаючи термінових заходів від України. Протягом тривалого часу економічний розвиток країни супроводжувався недооцінкою захисту довкілля, що заважало досягненню сталого розвитку.

Екологічне законодавство України становить систему нормативно-правових актів, які регулюють використання природних ресурсів та охорону навколишнього середовища. Система є прогресивною і була створена з використанням досвіду світових лідерів у галузі екологічного права.

Міжнародне екологічне право є ключовим для вирішення екологічних проблем в Україні через глобальний характер цих проблем та міжнародні зобов'язання країни. Україна є учасником численних міжнародних конвенцій та угод, що регулюють

охорону природи та використання природних ресурсів. Здійснення зобов'язань вимагає внутрішнього законодавчого врегулювання та врахування міжнародної практики.

4.2. Впровадження стандартів з охорони навколишнього середовища

До законодавства, що регулює охорону навколишнього середовища, в рамках Угоди про асоціацію України з Європейським Союзом, внесено низку нормативних актів ЄС. Серед них зазначаються:

1. **Енергетична стратегія України на період до 2035 року:** Вона встановлює високі стандарти екологічної відповідальності для держави та енергетичних компаній, вимагаючи дотримання високих екологічних стандартів у всіх етапах енергетичного процесу.
2. **Національний план скорочення викидів від великих установок, що спалюють:** Зазначено, що до 2028 року викиди пилу та оксидів сірки повинні скоротитися відповідно на 40 та 20 разів, а до 2033 року викиди оксидів азоту мають зменшитися у чотири рази відповідно до директиви 2010/75/ЄС.
3. **Національна стратегія управління відходами на період до 2030 року:** Передбачено розробку національних та регіональних планів управління відходами.
4. **Закон України «Про оцінку впливу на довкілля»:** Встановлено новий механізм оцінки впливу планової діяльності, що становить підвищену екологічну небезпеку, та передбачено відповідальність за невиконання положень.
5. **Закон України «Про внесення змін до Бюджетного кодексу України»:** Змінено розподіл коштів від екологічного податку між центральним та регіональними бюджетами.
6. **Закон України «Про внесення змін до Податкового кодексу України та деяких інших законодавчих актів України щодо покращення адміністрування та перегляду ставок окремих податків та зборів»:** Внесено зміни, збільшено ставку екологічного податку за викиди CO2.

7. **Наказ Державної служби статистики України «Про затвердження форми державного статистичного спостереження №2-ТП (повітря)»:** Затверджено вдосконалену форму звіту про викиди забруднюючих речовин та парникових газів від стаціонарних джерел викидів.

ВИСНОВКИ

У ході дослідження були розроблені моделі виявлення атак на мережі, використовуючи персептрон та нейронну мережу з алгоритмом зворотного розповсюдження помилки, що базується на багатошаровому персептроні.

Для виконання завдання магістерської роботи був проведений збір та аналіз даних для моделювання. Також були розроблені алгоритми передпроцесингу даних для виявлення атак. Модель нейронної мережі була синтезована, і проведено її навчання. Результати дослідження були використані для проведення аналізу придатності синтезованої моделі.

Розроблено алгоритми підготовки даних для навчання нейронної мережі для наступного її навчання.

Дослідження вказує на те, що модель аналізу мережевих потоків може бути ефективно використана для виявлення брутфорс-атак на FTP і SSH-сервіси, Dos-атак з використанням HOIC, HULK, а також активності ботнетів. Крім того, ця модель може слугувати інструментом для виявлення модулів для систем управління інцидентами безпеки (SIEM).

Синтезована модель для ідентифікації SQL-ін'єкцій має високу точність, приблизно 90%, і здатна виявляти веб-атаки, такі як брандмауери веб-додатків.

Майбутні напрямки роботи включають розробку інтелектуальних моделей для виявлення різноманітних атак на мережеві сервіси, централізацію цих моделей у єдиному програмному забезпеченні та візуалізацію їх роботи.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Звіт про розслідування витоків даних за 2019 р. // Verizon. – 2019.

2. Ентоні Дж. П. Штучні нейронні мережі з використанням багатошарового перцептрону [електронний ресурс] / П. Ентоні Дж., Донг Су Кім - Режим доступу до ресурсу: https://www.cse.unsw.edu.au/~cs9417ml/MLP2 /index.html.

3. Еклі Д.Х. Алгоритм навчання машин Больцмана / Д.Х. Еклі, Дж.Е. Хінтон, Т.Дж. Сейновський. // Когнітивна наука. - 1985. - № 9. - С.

4. Буллінг в епоху цифрових технологій: критичний огляд та метааналіз досліджень кіберзалякування серед молоді / Р. М. Ковальський, Г. В. Джуметті, А. Н. Шредер, М. Р. Латтаннер. // Психологічний вісник. - 2014. - № 140. - С. 1073.

5. Автоэнкодер слів/К. Лю, К. Ченг, Ж. Ліу, Д. Ліу. // Нейрокомп'ютінг. - 2014. - № 139. - С.

6. Темні мотиви онлайн: аналіз технологій, що перетинаються, використовуються кіберзлочинцями і терористичними організаціями [Електронний ресурс] // TrendMicro. – 2016. – Режим доступу до ресурсів:https://www.trendmicro.com/vinfo/us/security/новини/кіберзлочинність та цифрові загрози/технології-кіберзлочинці-і-терористичні організації.

7. Кук'є М. Дослідження: Хакери атакують кожні 39 секунд [електронний ресурс] / Мішель Кук'є // Інженерна школа А. Джеймса Кларка. – 2007 р. – Режим доступу до ресурсу: https://eng.umd.edu/news/story/study-hackers-attack-every-39-секунди.

8. Deep Belief Networks [електронний ресурс] // DeepLearning – режим доступу до ресурсу: http://deeplearning.net/tutorial/DBN.html.

9. Gartner прогнозує, що світові витрати на інформаційну безпеку перевищать $124 млрд 2019 року [електронний ресурс] // Gartner. – 2018 р. – Режим доступу до ресурсу: https://www.gartner.com/en/newsroom/press-releases/2018-08-15-gartner-forecasts-worldwide-information-security-spending-to-exceed- 124 – мільярдів у 2019 році.

10. Глобальна шкода від кіберзлочинності, за прогнозами, до 2021 року сягне $6 трлн щорічно [електронний ресурс] // Cybercrime Magazine. – 2020 р. – Режим доступу до ресурсу: https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/.

11. Хасті Т. Елементи статистичного навчання: інтелектуальний аналіз даних, логічні висновки та прогнозування / Т. Хасті, Р. Тібширані, Дж. Фрідман. - Нью-Йорк: Спрінгер, 2009. - 764 с. - (два).

12. Хопфілд Дж. Дж. Нейронні мережі та фізичні системи з колективними обчислювальними здібностями, що розвиваються / Хопфілд. //Анали Національної академії наук. - 1982. - № 79. - С.

13. LSTM: Пошукова космічна одіссея / К. Грефф, Р. К. Шрівастава, Дж. Кутнік, Б. Р. Стойнебринк., 2015. - 12 с.

14. Мерфі К. П. Машинне навчання: ймовірнісна перспектива / Кевін Мерфі. - Кембридж: MIT Press, 2012. - 247 с.

15. Новачок Е. Uber заплатив хакерам за видалення вкрадених даних 57 мільйонів людей [Електронний ресурс] / Ерік Ньюкомер // Bloomberg: Кібербезпека. – 2017 р. – Режим доступу до ресурсу: https://www.bloomberg.com/news/articles/2017-11-21/uber-concealed-cyberattack-that-expose-57-million-people-s-data.

16. овасп. SQL-ін'єкція [електронний ресурс] / OWASP – режим доступу до ресурсу: https://owasp.org/www-community/attacks/SQL_Injection.

17. Роуз М. SOAR (Оркестрація безпеки, автоматизація та реагування) [електронний ресурс] / Маргарет Роуз // SearchSecurity - режим доступу до ресурсу: https://searchsecurity.techtarget.com/definition/SOAR.

18. Понемон Л. Що нового у звіті про витрати на витік даних у 2019 році [Електронний ресурс] / Ларрі Понемон // Security Intelligence. – 2019 р. – Режим доступу до ресурсу: https://securityintelligence.com/posts/whats-new-in-the-2019-cost-of-a-data-breach-report/.

19. Загальна кількість веб-сайтів [електронний ресурс]// Інтернет-статистика в реальному часі – режим доступу до ресурсу: https://www.internetlivestats.com/total-number-of-websites/.

20. Що таке SQL-ін'єкція (SQLi) та як її уникнути [електронний ресурс] // Acunetix. – Режим доступу до ресурсів: https://www.acunetix.com/websitesecurity/sql-injection/.

21. Керівництво WEKA для версії 3-9-3/[Р. Букерт, Е. Франк, М. Хол та ін.]. -Гамільтон: Університет Вайкато.

22. Вільямс А. Підвищення IT-безпеки за допомогою управління вразливістю [Електронний ресурс]/А. Вільямс, М. Ніколетт. – 2015 р. – Режим доступу до ресурсу: https://www.gartner.com/en/documents/480703/improve-it-security-with-vulnerability-management.

23. Хайкін С. Нейронні мережі: комплексний фундамент/Саймон Хайкін. - Москва: Вільямс, 2006. - 1104 с.

24. Наслідки кібератак для малого та середнього бізнесу [електронний ресурс]// Hi-Tech. – 2015. – Режим доступу до ресурсу: https://hi-tech.ua/blog/posledstviya-kiberatak-dlya-malogo-i-srednego-biznesa/.

25. Суміт С. Комплексний посібник з згорткових нейронних мереж [Електронний ресурс] / Саха Суміт - Режим доступу до ресурсів: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way - 3б2б1164a53.

26. Источник: беседа с Bing, 10.12.2023

27. (1) Екологічне законодавство – Департамент екології та природніх ресурсів .... https://deplv.gov.ua/ekologichne-zakonodavstvo/.

28. (2) Презентація на тему "Міжнародне екологічне право. Екологічне .... https://naurok.com.ua/prezentaciya-na-temu-mizhnarodne-ekologichne-pravo-ekologichne-zakonodavstvo-ukra-ni-ponyattya-informaciynogo-suspilstva-ta-stalogo-rozvitku-285645.html.

29. (3) Екологічні права та обов'язки громадян України — Вікіпедія. https://bing.com/search?q=%d0%b5%d0%ba%d0%be%d0%bb%d0%be%d0%b3%d1%96%d1%87%d0%bd%d0%b5+%d0%b7%d0%b0%d0%ba%d0%be%d0%bd%d0%be%d0%b4%d0%b0%d0%b2%d1%81%d1%82%d0%b2%d0%be+%d1%83%d0%ba%d1%80%d0%b0%d1%97%d0%bd%d0%b8.

30. (4) Законодавство про екологію: що змінилося протягом 2022 року. https://mind.ua/openmind/20251329-zakonodavstvo-pro-ekologiyu-shcho-zminilosya-protyagom-2022-roku.

31.

ДОДАТОК А

Реалізація нейромережевої моделі виявлення мережевих атак з урахуванням аналізу мережевих атрибутів.

Тест.java

```
Імпортуватиweka.classifiers.functions.MultilayerPerceptron;
Імпортуватиweka.core.Debug;
Імпортуватиweka.core.Instances;
Імпортуватиweka.filters.Фільтр;
Імпортуватиweka.filters.unsupervised.attribute.Normalize;
Імпортуватиweka.filters.unsupervised.attribute.NumericToNominal;

громадський клас Тест{

 громадський статичний Фінал Нитка ШЛЯХ НАБОРУ ДАНИХ"=""*.arff";
 громадський статичний Фінал Нитка ШЛЯХ МОДЕЛІ"=""*.bin";
 громадський статичний Фінал Нитка ШЛЯХ ПЕРЕВІРКИ"=""*.arff";

 громадський статичний порожній Основний(Нитка[]аргументи)спис Виняток{
 Класифікуватиекземпляр();
}

 статичний порожній Генератемодель()спис Виняток{
 Генератор моделей мг"="новий Генератор моделей();

 Примірники набір даних"="мг.завантажити набір даних(ШЛЯХ НАБОРУ ДАНИХ);
 Система.зовні.друкувати(«Завантажений набір даних»);
 Фільтр numToNominalFilter"="новий числовий-номінальний();
 Нитка[]параметри"="новий Нитка[два];
параметри[0"=""-P";
параметри[1"=""два";
 numToNominalFilter.ВстановитиОпції(Параметри);
 numToNominalFilter.setInputFormat(набір даних);
набір даних =Фільтр.використовуватиФільтр(набір даних, numToNominalFilter);
 Система.зовні.друкувати(«Застосовано цифровий фільтр до номінального»);

 Фільтр фільтр"="новий Нормалізувати();

 //поділяємо набір даних, щоб навчити набір даних 80% і протестувати набір даних 20%
 внутрішній розмір поїзда"="внутрішній)Математика.круглий(набір даних.кількість
екземплярів()*0,9);
 внутрішній testSize"="набір даних.кількість екземплярів() - Розмір поїзда;

 набір даних.Випадковий(новийНалагоджувати.Випадковий(1));// якщо ви прокоментуєте цей
рядок, точність моделі знизиться
 // Від 96,6% до 80%
 Система.зовні.друкувати(«Випадковий успіх набору даних»);
```

```
 // Нормалізуємо набір даних
 фільтр.setInputFormat(набір даних);
 Примірники набір даних нор"="Фільтр.використовуватиФільтр(Набір даних, фільтр);
 Система.зовні.друкувати("Нормалізація застосованого фільтра");

 Примірники набір даних поїзда"="новий Примірники(ні набір даних,0, Розмір поїзда);
 Примірники тестовий набір даних"="новий Примірники(набір даних, trainSize, testSize);

 //Створюємо класифікатор з набором навчальних даних
 Багатошаровий персептрон A-H-A= (Багатошаровий персептрон)мг.buildClassifier(Набір
навчальних даних);
 Система.зовні.друкувати(«Класифікатор успішно побудований»);

 // Оцінюємо класифікатор за допомогою тестового набору даних
 Нитка резюме оцінки"="мг.ставкаМодель(Енн, набір даних поїзда, набір тестових даних);
 Система.зовні.друкувати("Оцінка:"+ резюме оцінки);

 //Зберігаємо модель
 мг.зберегти модель(Рік, ШЛЯХ МОДЕЛІ);
}

 статичний порожній Класифікуватиекземпляр()спис Виняток{
 Генератор моделей мг"="новий Генератор моделей();

 Примірники набір даних"="мг.завантажити набір даних(ПЕРЕВІРКА ШЛЯХУ);
 Система.зовні.друкувати(«Набір тестів завантажений»);

 Фільтр numToNominalFilter"="новий числовий-номінальний();
 Нитка[]параметри"="новий Нитка[два];
параметри[0"=""-P";
параметри[1"=""два";
 numToNominalFilter.ВстановитиОпції(Параметри);
 numToNominalFilter.setInputFormat(набір даних);
набір даних =Фільтр.використовуватиФільтр(набір даних, numToNominalFilter);

 Фільтр фільтр"="новий Нормалізувати();
 фільтр.setInputFormat(набір даних);
набір даних =Фільтр.використовуватиФільтр(Набір даних, фільтр);

 МодельКласифікатор клс"="новий МодельКласифікатор();
 /*
* Validator[] validators = new Validator[] { new Validator("Benigno"), new
* Валідатор("FTP-BruteForce"), новий Валідатор("SSH-BruteForce") };
*/
 Валідатор[]валідатори"="новий Валідатор[] {новий Валідатор(«Доброякісний»),новий
Валідатор(«Інфільтрація»),
 новий Валідатор(«Робот»)};

 для(Валідатор в :валідатори) {
 в.цінності"="новий внутрішній[валідатори.довжина];
}
```

```
 для(внутрішній я"="0; та <набір даних.кількість екземплярів(); е++) {
 Нитка ім'я класу"="клс.ранжувати(набір даних.брати(і), ШЛЯХ МОДЕЛІ);
 Нитка клас за замовчуванням"="набір даних.брати(Я).рядкове значення(66);
 Валідатор.оновленняВалідація(Валідатори, defaultClass, ім'я класу);
}
 подвійний відносна помилка"="0;
 для(внутрішній я"="0; та <валідатори.довжина; е++) {
 для(внутрішній Дж."="0; j<валідатори.довжина; j++) {
 якщо(Я! = j)
відносна помилка += валідатори[i].цінності[Дж];
}
}
 Валідатор.друкРезультат(валідатори);
 Система.зовні.друкувати(«Відносна помилка:»+ (відносна помилка /набір даних.кількість
екземплярів()*100) +"%");
}

}
```

МодельГенератор.java

```
Імпортуватиjava.util.logging.Level;
Імпортуватиjava.util.logging.Logger;
Імпортуватиweka.classifiers.Classifier;
Імпортуватиweka.classifiers.evaluation.Evaluation;
Імпортуватиweka.classifiers.functions.MultilayerPerceptron;
Імпортуватиweka.core.Instances;
Імпортуватиweka.core.SerializationHelper;
Імпортуватиweka.core.converters.ConverterUtils.DataSource;

громадський клас Генератор моделей{

 громадський Примірники завантажити набір даних(Нитка шлях) {
 Примірники набір даних"="нульовий;
 намагатися{
набір даних =Джерело даних.читати(Шлях);
 якщо(набір даних.індекс класу() == -1) {
 набір даних.setClassIndex(набір даних.numAttributes() -1);
}
}брати(Виняток колишній) {
 Реєстратор.getLogger(Генератор
моделей.клас.getName()).записувати(Рівень.СИЛЬНИЙ,нульовий, Колишній);
}

 повернутисянабір даних;
}

 громадський Класифікатор buildClassifier(Примірники набір даних поїзда) {
 Багатошаровий персептрон я"="новий Багатошаровий персептрон();
```

```
 //m.setGUI(істина);
 //m.setValidationSetSize(0);
 // m.setBatchSize("100");
 я.setLearningRate(0,5);
 //m.setSeed(0);
 //m.setMomentum(0.2);
 я.setTrainingTime(50);//пори року
 //m.setNormalizeAttributes(true);
 намагатися{
 я.buildClassifier(Набір навчальних даних);

}брати(Виняток колишній) {
 Реєстратор.getLogger(Генератор
моделей.клас.getName()).записувати(Рівень.СИЛЬНИЙ,нульовий, Колишній);
}
 повернутисям;
}

 громадський Нитка ставкаМодель(Класифікатор Модель,Примірники набір даних
поїзда,Примірники тестовий набір даних)спис Виняток{
 Оцінка оцінка"="нульовий;
 намагатися{
 // Оцінюємо класифікатор за допомогою тестового набору даних
оцінка =новий Оцінка(Набір навчальних даних);
 оцінка.ставкаМодель(Модель, тестовий набір даних);
}брати(Виняток колишній) {
 Реєстратор.getLogger(Генератор
моделей.клас.getName()).записувати(Рівень.СИЛЬНИЙ,нульовий, Колишній);
}
 Нитка результат"="оцінка.toSummaryString("Короткий зміст",істинний) +"\n"
+оцінка.toClassDetailsString(«Детальна точність за класами») +"\n"
+оцінка.toMatrixString(«Матриця плутанини»);
 повернутисярезультат;
}

 громадський порожній зберегти модель(Класифікатор Модель,Нитка шлях моделі) {

 намагатися{
 Помічник із серіалізації.написати(Модельний шлях, модель);
}брати(Виняток колишній) {
 Реєстратор.getLogger(Генератор
моделей.клас.getName()).записувати(Рівень.СИЛЬНИЙ,нульовий, Колишній);
}
}

}
```

МодельКласифікатор.java

```
Імпортуватиjava.util.ArrayList;
Імпортуватиjava.util.logging.Level;
Імпортуватиjava.util.logging.Logger;
Імпортуватиweka.classifiers.Classifier;
Імпортуватиweka.classifiers.functions.MultilayerPerceptron;
Імпортуватиweka.core.Attribute;
Імпортуватиweka.core.DenseInstance;
Імпортуватиweka.core.Instances;
Імпортуватиweka.core.Instance;
Імпортуватиweka.core.SerializationHelper;

громадський клас МодельКласифікатор{

 приватний ListArray<Нитка>класVal;
 приватний Примірники необроблені дані;

 громадський МодельКласифікатор() {
клас Val =новий ListArray<Нитка>();
 класVal.додати("Твій клас");
}

 громадський Примірники створити екземпляр(подвійний довжина пелюстки,подвійний ширина
пелюстки,подвійний результат) {
 необроблені дані.звичайно();
 подвійний[]екземпляр значення1"="новий подвійний[ ] { Довжина пелюстки, ширина
пелюстки,0};
 необроблені дані.додати(новий Щільний екземпляр(1.0, Значення екземпляра1));
 повернутисянеоброблені дані;
}

 громадський Нитка ранжувати(приклад знімок,Нитка шлях) {
 Нитка результат""Без рейтингу!!";
 Класифікатор клс"="нульовий;
 намагатися{
cls = (Мультишаровий перцептрон)Помічник із серіалізації.читати(Шлях);
результат =класVal.брати((внутрішній)клс.класифікуватиекземпляр(Інст));
}брати(Виняток колишній) {

Реєстратор.getLogger(МодельКласифікатор.клас.getName()).записувати(Рівень.СИЛЬНИЙ,нульов
ий, Колишній);
}
 повернутисярезультат;
}

 громадський Примірники отримати екземпляр() {
 повернутисянеоброблені дані;
}
```

```
}
```

Валідатор.java

```
Імпортуватиjava.util.Массиви;

громадський клас Валідатор{
 громадський Нитка ім'я;
 громадський внутрішній[]цінності;

 громадський Валідатор(Нитка ім'я) {
 що.ім'я= ім'я;
}

 громадський статичний порожній оновленняВалідація(Валідатор[]він прибуває,Нитка
визначення,Нитка ім'я класу) {
 внутрішній поз1"="0;
 внутрішній позиція 2"="0;
 для(внутрішній я"="0; та <він прибуває.довжина; е++) {
 якщо(Обр[я].ім'я.це те саме, що(за замовчуванням) && позиція1 ==0) {
позиція1 = я;
}
 якщо(Обр[я].ім'я.це те саме, що(ім'я класу) && pos2 ==0) {
позиція2 = я;
}
}
обр [поз1].цінності[поз2] +=1;
}

 громадський статичний порожній друкРезультат(Валідатор[]він прибуває) {
 для(внутрішній я"="0; та <він прибуває.довжина; е++) {
 Система.зовні.друкувати(Обр[я].ім'я+":"+Матриці.(обр[я].цінності));
}
}
}
```

збирач даних.ру

```
Імпортуватисистема
Імпортуватиcsv
Імпортуватиоперацийна система
кількість =0
повітря = []
шлях = os.path.dirname(os.path.abspath(__файл__))

файли = [1.csv','2.csv']
результат =Відкрити(шлях +'/**/test-o.csv',"ш",нова лінія"="'')
wtr = csv.writer (результат)
header_exist =Брехня
```

```
дляжвфайли:
 з Відкрити(шлях + ж,"р")якджерело:
rdr = csv.reader (джерело)
 длядж, рв перераховувати(РДР):
 якщой ==0 Цезаголовок_існує:
 Продовжувати
header_exist =істинний
піз = []
фільтр =Брехня
 дляя Вінв перераховувати(р):
 якщоя ==0 Цедж! "="0 Це ні(Ел =="80" абовін =="443" абовін =="445" абовін =="53" абовін
=="21" абовін =="8080"):
фільтр =істинний
 зламати
 якщоЯ! "="два ЦеЯ! "="16 ЦеЯ! "="17 ЦеЯ! "="38 ЦеЯ! "="39 ЦеЯ! "="44 ЦеЯ! "="57 ЦеЯ!
"="58 ЦеЯ! "="59 ЦеЯ! "="60 ЦеЯ! "="61 ЦеЯ! "="62 ЦеЯ! "="70:
res.append(ел)
 якщоя ==79 Цедж! "="0:
 якщоелектронна поштані вповітря:
arr.append(ел)
 якщо ніфільтр:
wtr.writerow (рез)
рахувати +=1
 друкувати(розказувати,кінець""\р")
джерело.закрити()
результат.закрити()
друкувати(Обр.)
кількість =0
сортований масив = {}
дляавповітря:
відсортований_масив[а] = []

default_header = []

з Відкрити(шлях +'/**/test-o.csv',"р")як файл:
дані = csv.leitor(файл)
 дляя пішовв перераховувати(дані):
 якщоя ==0:
default_header = d

 дляавповітря:
 якщод[66] == а:
sorted_array[а].append(d)
рахувати +=1
 друкувати(розказувати,кінець""\р")

дляЦевsorted_array.keys():
 друкувати(з +":"+вул.(повільний(сортований_масив[и])))

кількість =0
```

```
file_to_generation =Відкрити(
шлях +'/інфільтрація/learning.csv',"ш",нова лінія"="'')
file_for_test1 =Відкрити(шлях +'/**/test-1.csv',"ш",нова лінія"="'')
file_for_test2 =Відкрити(шлях +'/**/test-2.csv',"ш",нова лінія"="'')
w1 = csv.writer(файл_для_генерації)
w2 = csv.writer(file_for_test1)
w3 = csv.writer(file_for_test2)
w1.writerow(default_header)
w2.writerow(default_header)
w3.writerow(default_header)
дляЦевsorted_array.keys():
 дляя Вінв перераховувати(сортований_масив[s]):
 якщота <повільний(сортований_масив[s])/два:
w1.writerow(ел)
 Еліфя >=повільний(упорядкований_масив[s]) *0,5 Цета <повільний(упорядкований_масив[s])
*0,75:
w2.writerow(ел)
 інший:
w3.writerow(ел)
рахувати +=1
 друкувати(розказувати,кінець"=""\p")
file_to_generation.close()
file_for_testing1.close()
file_for_testing2.close()
```

ДОДАТОК Б

Реалізація моделі нейронної мережі для ідентифікації SQL-ін'єкцій

rede.js

```
константасинаптичний"="вимагати("синаптичний");
константаcsv"="вимагати("парсер csv");
константашлях"="вимагати(шлях);
константафс"="вимагати("ФС");
константаАрхітектор"="синаптичний.Архітектор;
константаТренер"="синаптичний.Тренер;
константаКласифікатор"="вимагати("./класифікатор");

варфазингSet"="Класифікатор.отриматиDataSet("ін'єкція.txt");
вардоброякісний набір"="Класифікатор.отриматиDataSet("чистий.txt");

вармережа"="новийАрхітектор.Персептрон(12,6,1);
вармій тренер"="новийТренер(мережа);

// Нехай FinalSet = [];
// for (let i = 0; i < fuzzingSet.length; i++) {
// якщо (я % 2 == 0)
//finalSet.push({
// введення: fuzzingSet[i],
// Висновок: [1]
//});
// інший
//finalSet.push({
// Введення: benignSet [i],
// Висновок: [0]
//});
// }

функціяgetInjectionDataFromCsv(Шлях до файлу) {
повернутисяновийОбіцяти(вирішувати=>{
ЗалишитиРезультати"="
фс.створитиReadStream(Шлях до файлу)
.трубка(csv())
.він("дані",дані=>Результати.штовхати(дані))
.він("кінець", ()=>{
ЗалишитиsqliT"="Результати
.фільтр(електронна пошта=>електронна пошта.тип_атаки=="sql")
.карта(електронна пошта=>Класифікатор.бінарна візуалізація(електронна пошта.корисне
навантаження))
.конкат(фазингSet);
Залишитидоброякісний"="Результати
.фільтр(електронна пошта=>електронна пошта.тип_атаки=="нормальний")
.карта(електронна пошта=>Класифікатор.бінарна візуалізація(електронна пошта.корисне
навантаження))
```

```
.конкат(доброякісний набір);

вармійСет"="
...доброякісний.карта(щасливий=>{
повернутися{
входи: зміст,
виходи:[0]
};
}),
...sqliT.карта(щасливий=>{
повернутися{
входи: зміст,
виходи:[1]
};
})
];
Залишитирезультат"="мій тренер.тренуватися(мійСет, {
лапки:0,02,
ітерації:200,
помилка:0,005,
перемішати:істинний,
Історичний:20
});

консоль.записувати(результат);

консоль.записувати(«Доброякісний поїзд:»,доброякісний.довжина);
консоль.записувати("Набір поїздів SQLi:",sqliT.довжина);
вирішувати(результат);
});
});
}

функціяконтрольний тест() {
консоль.записувати(
мережа.активувати(Класифікатор.бінарна візуалізація(«/Привіт/я добрий/вибрати з»))
);
консоль.записувати(
мережа.активувати(
Класифікатор.бінарна візуалізація(
"/en/brand/km-by-lange-en/?category=спідниця міді exec master..xp_cmdshell
'ipconfig+/all'"
)
)
);
консоль.записувати(
мережа.активувати(
Класифікатор.бінарна візуалізація(
"'union (виберіть NULL, NULL, NULL, NULL, NULL, (виберіть @@version)) --"
)
)
```

```
);
консоль.записувати(мережа.активувати([1,1,1,1,1,1,1,1,1,1,1,1]));

ЗалишитиконтрольДоброякісний"="Класифікатор.отриматиDataSet("control-b.txt");
ЗалишитиконтрольІн'єкція"="Класифікатор.отриматиDataSet("control-i.txt");

для(Залишитися"="0;я<контрольДоброякісний.довжина;я++) {
Залишитир"="мережа.активувати(контрольДоброякісний[я]);
консоль.записувати(«Доброякісний:»,р);
}

для(Залишитися"="0;я<контрольІн'єкція.довжина;я++) {
Залишитир"="мережа.активувати(контрольІн'єкція[я]);
консоль.записувати(«Ін'єкція:»,р);
}

консоль.записувати(
«Контрольний зразок:»,
контрольДоброякісний.довжина+контрольІн'єкція.довжина,
контрольДоброякісний.довжина,
контрольІн'єкція.довжина
);
}

getInjectionDataFromCsv(
шлях.об'єднати("HttpParamsDataset","carga_trem.csv")
).потім(Модель=>{
ЗалишитиРезультати"="
фс.створитиReadStream(шлях.об'єднати("HttpParamsDataset","carga_teste.csv"))
.трубка(csv())
.він("дані",дані=>Результати.штовхати(дані))
.він("кінець", ()=>{
ЗалишитиsqliT"="Результати
.фільтр(електронна пошта=>електронна пошта.тип_атаки=="sql")
.карта(електронна пошта=>Класифікатор.бінарна візуалізація(електронна пошта.корисне
навантаження));

Залишитидоброякісний"="Результати
.фільтр(електронна пошта=>електронна пошта.тип_атаки=="нормальний")
.карта(електронна пошта=>Класифікатор.бінарна візуалізація(електронна пошта.корисне
навантаження));

консоль.записувати(
«Доброякісний набір тестів:»,
доброякісний.нарізати(0,Математика.підлога(доброякісний.довжина/два)).довжина
);
консоль.записувати(
«Набір тестів SQLi:»,
sqliT.нарізати(0,Математика.підлога(sqliT.довжина/два)).довжина
);
Залишитирезультат"="успіх_б:0,помилка_б:0,успіх_я:0,помилка_я:0};
```

```
для(ЗалишитизаписувативsqliT.нарізати(0,Математика.підлога(sqliT.довжина/два))) {
Залишитир"="мережа.активувати(записувати)[0];
якщо(р<0,5)результат.помилка_я++;
іншийрезультат.успіх_я++;
}
для(Залишитизаписувативдоброякісний.нарізати(0,Математика.підлога(доброякісний.довжина/д
ва))) {
Залишитир"="мережа.активувати(записувати)[0];
якщо(р>=0,5)результат.помилка_б++;
іншийрезультат.успіх_б++;
}
консоль.записувати("Результат випробувань:",результат);

консоль.записувати(
«Доброякісний контрольний набір:»,
доброякісний.нарізати(Математика.підлога(доброякісний.довжина/два),доброякісний.довжина)
.довжина
);
консоль.записувати(
«Набір елементів керування SQLi:»,
sqliT.нарізати(Математика.підлога(sqliT.довжина/два),sqliT.довжина).довжина
);
результат"="успіх_б:0,помилка_б:0,успіх_я:0,помилка_я:0};

для(ЗалишитизаписувативsqliT.нарізати(
Математика.підлога(sqliT.довжина/два),
sqliT.довжина
)) {
Залишитир"="мережа.активувати(записувати)[0];
якщо(р<0,5)результат.помилка_я++;
іншийрезультат.успіх_я++;
}
для(Залишитизаписувативдоброякісний.нарізати(
Математика.підлога(доброякісний.довжина/два),
доброякісний.довжина
)) {
Залишитир"="мережа.активувати(записувати)[0];
якщо(р>=0,5)результат.помилка_б++;
іншийрезультат.успіх_б++;
}
консоль.записувати("Результат випробувань:",результат);
//controlTest()
});
});

вармійСет"="
...доброякісний набір.карта(щасливий=>{
повернутися{
входи: зміст,
виходи:[0]
```

```
};
}),
...фазингSet.карта(щасливий=>{
повернутися{
входи: зміст,
виходи:[1]
};
})
];
```

класифікатор.js

```
константафс"="вимагати("ФС");
константашлях"="вимагати(шлях);

константаКласи"="
"'",
"СТВОРЮВАТИ",
"ВИДАЛИТИ З",
«ОСІННІЙ СТІЛ»,
«ВСТАВИТИ В»,
"ЄДНІСТЬ",
"І",
"АБО",
"--",
" ",
"В",
"ЕКСП"
].карта(електронна пошта=>електронна пошта.для нижнього регістру());

функціябінарна візуалізація(вул.) {
повернутисяКласи.карта(стандартний=>(вул..для нижнього
регістру().включає(стандартний)?1:0));
}

функціяотриматиDataSet(ім'я файлу) {
вардані"="фс.читанняфайлесинк(шлях.об'єднати(__ім'я_каталогу,ім'я файлу), {
кодифікація:"utf-8"
});
дані"="дані.ділити("\n");
для(Залишитися"="0;я<дані.довжина;я++) {
дані[я"="бінарна візуалізація(дані[я]);
}
повернутисядані;
}

модуль.експорт"="отриматиDataSet,бінарна візуалізація};
```