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: Інструменти для виявлення загроз в мережах з використанням Python

Виконавець: Ростислав Фурман

Керівник: д.т.н., проф., зав каф. Михайло СТЕПАНОВ

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

Нормоконтролер: д.т.н., проф., зав каф. Михайло СТЕПАНОВ

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: Кібербезпеки, комп'ютерної 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 Фурмана Ростислава Михайловича

1. Тема: Інструменти для виявлення загроз в мережах з використанням Python.
   затверджена наказом ректора від «15» 09 2023 № 1814/ст.
2. Термін виконання з 16.10.2023 по 31.12.2023
3. Вихідні дані: сучасний стан Python в сфері захисту інформаційних систем, його перевага у машинному навчанні..
4. Зміст пояснювальної Проектування та впровадження моделі для захисту від шкідливих даних.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

№ п/п	Етапи виконання кваліфікаційної роботи	Термін виконання етапів	Примітка
1.	Уточнення постановки задачі	16.10.23 – 20.10.23	Виконано
2.	Аналіз наявних засобів для шифрування електронних повідомлень	20.10.23 –26.10.23	Виконано
3.	Аналіз літературних джерел	26.10.23 –01.11.23	Виконано
4.	Аналіз моделей машинного навчання	01.11.23 –08.11.23	Виконано
5.	Ознайомлення з технологіями для розробки програмного забезпечення	08.11.23 –17.11.23	Виконано
6.	Проектування алгоритмів основних функції програми	17.11.23 –22.11.23	Виконано
7.	Програмна реалізація розроблених алгоритмів	22.11.23 30.11.23	Виконано
8.	Тестування розробленого програмного засобу	30.11.23 –05.12.23	Виконано
9.	Оформлення пояснювальної записки та графічного матеріалу	05.12.23 –22.12.23	Виконано

1. Консультанти з окремих розділів

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

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

Здобувач вищої освіти Ростислав Фурман

(підпис, дата)

Керівник Михайло СТЕПАНОВ

(підпис, дата)

РЕФЕРАТ

Кваліфікаційна робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, додатків і має 65 сторінки основного тексту, 4 таблиці, 5 рис., 18 сторінок додатків. Список використаних джерел містить 29 найменувань і займає 3 сторінки. Загальний обсяг роботи 88 сторінок.

Метою роботи є розробка та реалізація моделі машинного навчання, призначеного для захисту від атак через підозрілі посилання.

В роботі розроблено алгоритм та програмне забезпечення для впровадження та навчання моделі.

Розроблений алгоритм та програмне забезпечення відноситься до галузі кібербезпеки і може бути використане для підвищення рівня захищеності.

Ключова задача - виявлення та класифікація URL-адрес, що можуть бути потенційно шкідливими або підозрілими. Створено набір алгоритмів, які ефективно аналізують та розпізнають ознаки шкідливості в URL-адресах, використовуючи різні методи машинного навчання, такі як Decision Tree, AdaBoost, Logistic Regression, Gaussian Naive Bayes та KNN.

Крім навчання моделей, приділили значну увагу підвищенню їх точності через тонку настройку гіперпараметрів за допомогою методу GridSearchCV. Це дозволило вибрати найкращі параметри для кожної моделі, значно підвищивши їх продуктивність.

Результати дипломного проектування рекомендується використовувати для забезпечення додаткової безпеки в підприємствах, державних установах , для яких важливим є питання захисту та в освітньому процесі НАУ.

Ключові слова: МЕРЕЖА, ШИФРУВАННЯ, SURICATA, ПРАВИЛА, РЕГІСТР ЗСУВУ, ФАЄРВОЛ,МАШИННЕ НАВЧАННЯ,

Зміст

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

HTTP (Hypertext Transfer Protocol) – Протокол передачі гіпертексту;

IDS (Intrusion Detection System) – Системи виявлення вторгнень;

IPS (Intrusion Prevention System) – Система попередження вторгнень;

LOF (Local Outlier Factor) – Локальний коефіцієнт викидів;

NAB (Numenta Anomaly Benchmark) – Показник аномалій;

NAC (Network access control) – Контроль доступу до мережі;

NBA (Network Behavior Analysis) – Аналіз поведінки мережі

ADS (Anomaly Detection System) – Системи виявлення аномалій;

DoS (Denial of Service) – Відмова в обслуговування;

DDoS (Distributed Denial of Service) – Розподілений DoS;

SUOD (Scalable Unsupervised Outlier Detection) – Масштабоване неконтрольоване виявлення викидів;

TCP (Transmission Control Protocol) – Протокол управління передачею даних;

TODS (Automated Time-series Outlier Detection System) – Автоматизована система машинного навчання;

ВСТУП

Сучасні мережі є життєво важливими для бізнесу, уряду та суспільства в цілому. Вони використовуються для передачі даних, доступу до Інтернету, управління обладнанням та надання інших важливих послуг. Однак ці мережі також є вразливими до різних загроз, таких як мережеві атаки, шкідливе програмне забезпечення та вразливості в мережевих пристроях.

Мережеві атаки можуть призвести до крадіжки даних, порушення конфіденційності, втрати продуктивності та навіть фінансових збитків. Шкідливе програмне забезпечення може пошкодити або знищити комп'ютери та мережі, а також призвести до крадіжки особистої інформації. Вразливості в мережевих пристроях можуть бути використані для отримання несанкціонованого доступу до мережі або її контролю.

За даними дослідження компанії Cybersecurity Ventures, у 2023 році очікується, що вартість глобальних збитків від кіберзлочинності досягне 10,5 трильйонів доларів США. Це означає, що кіберзлочинність є однією з найдорожчих проблем, з якими стикається світ сьогодні.

Для захисту мереж від цих загроз використовуються різні інструменти для виявлення загроз. Ці інструменти можуть використовуватися для моніторингу мережевого трафіку, виявлення шкідливого програмного забезпечення та виявлення вразливостей у мережевих пристроях.

Мова програмування Python є потужним і універсальним інструментом, який можна використовувати для розробки інструментів для виявлення загроз. Python має ряд переваг для розробки цих інструментів, включаючи:

1. Простота навчання та використання. Python має простий синтаксис, який легко освоїти навіть початківцям.

2. Велике і активне співтовариство розробників. Python має велике і активне співтовариство розробників, які створюють і

підтримують різні бібліотеки та інструменти, які можна використовувати для розробки інструментів для виявлення загроз.

3. Модульна архітектура. Python має модульну архітектуру, яка дозволяє легко розробляти та розширювати інструменти.

Ця тема є актуальною, оскільки вона стосується важливої проблеми захисту мереж від загроз. Використання Python для розробки інструментів для виявлення загроз може забезпечити ефективний і економічний спосіб захисту мереж.

Через глобалізацію, мережі стають все більш глобальними. Це ускладнює захист мереж від загроз, оскільки хакери можуть атакувати мережі з будь-якої точки світу.

Інтернет-орієнтованість. Більшість бізнесів і організацій сьогодні орієнтовані на Інтернет. Це означає, що їхні мережі є особливо вразливими до мережевих атак.

Розвиток мобільних пристроїв. Мобільні пристрої стають все більш популярними. Це означає, що мережі повинні бути захищені від загроз, пов'язаних з мобільними пристроями.

Загрози у мережах - це потенційні або реальні небезпеки, які можуть призвести до порушення конфіденційності, цілісності або доступності інформації та ресурсів в мережовому середовищі. Загрози можуть виникати як через технічні слабкості, так і через вразливості в людському чиннику.

Основні види загроз у мережах:

Малвара (Malware): Включає в себе віруси, троянські коні, черв'яки та інші шкідливі програми, які можуть завдати шкоди системі або вкрасти конфіденційні дані.

Атаки на введення (Input Attacks): Включають атаки, спрямовані на отримання невірного введення в програми чи системи для отримання несанкціонованого доступу.

Фішинг (Phishing): Атака, яка полягає в обмані користувачів і введенні їх в оманливі дії, такі як введення конфіденційної інформації.

Деніал сервісу (Denial of Service - DoS): Атаки, спрямовані на перешкоджання нормальному функціонуванню системи або мережі, зазвичай шляхом перевантаження ресурсів.

Атаки на вразливості (Exploits): Використання вразливостей у програмах чи операційних системах для отримання несанкціонованого доступу.

Атаки "людина посеред цього" (Man-in-the-Middle): Атаки, при яких атакуючий вставляється між двома комунікуючими сторонами, перехоплюючи або змінюючи передачу даних.

Методи виявлення загроз у мережах:

Виявлення аномалій (Anomaly Detection): Використовує аналіз відхилень від типового патерну поведінки для виявлення потенційно шкідливих дій.

Виявлення сигнатур (Signature-based Detection): Операції зіставлення входять до бази даних відомих сигнатур атак для виявлення вже відомих загроз.

Виявлення вразливостей (Vulnerability Scanning): Систематичне сканування системи для виявлення вразливостей та слабкостей.

Інтелектуальний аналіз поведінки (Behavioral Analysis): Моніторинг та аналіз поведінки користувачів та систем для виявлення аномальних дій.

Інспекція пакетів (Packet Inspection): Аналіз заголовків та вмісту пакетів даних для виявлення потенційно шкідливих або ненормальних патернів.

Штучний інтелект та машинне навчання (AI/ML): Використання алгоритмів машинного навчання для виявлення шаблонів та аномалій у мережах.

Огляд можливостей Python для виявлення загроз

Python - це потужна мова програмування, яка широко використовується для різних завдань, включаючи виявлення загроз. Вона має ряд переваг, які роблять її особливо придатною для цієї мети, зокрема:

1.Можливість аналізувати великі набори даних.

Python є мовою високого рівня, яка має потужні можливості для обробки даних. Це дозволяє використовувати його для аналізу великих наборів даних, таких як журнали веб-серверів, логови безпеки та дані мережевого моніторингу. Це може допомогти виявити потенційні загрози, такі як шкідливе програмне забезпечення, зловмисні атаки та аномалії в мережевому трафіку.

2. Можливість використовувати бібліотеки для виявлення загроз

Python має доступ до великої кількості бібліотек для виявлення загроз. Ці бібліотеки можуть допомогти виявити різні типи загроз, включаючи вразливості, шкідливе програмне забезпечення та зловмисні атаки. Наприклад, бібліотека OpenSCAP може використовуватися для виявлення вразливостей у системах та програмному забезпеченні. Бібліотека Snort може використовуватися для сканування мережі на предмет вразливостей та шкідливого програмного забезпечення. Бібліотека Suricata може використовуватися для виявлення атак на мережу.

3. Можливість автоматизувати завдання виявлення загроз

Python можна використовувати для автоматизації завдань виявлення загроз, таких як аналіз журналів, сканування мережі та розслідування інцидентів. Це може допомогти поліпшити ефективність процесу виявлення загроз. Наприклад, можна написати Python-скрипт для автоматичного аналізу журналів веб-серверів

на предмет потенційних загроз. Можна також написати Python-скрипт для автоматичного сканування мережі на предмет вразливостей та шкідливого програмного забезпечення.

Приклади використання Python для виявлення загроз

1.Аналіз журналів веб-серверів

Python можна використовувати для аналізу журналів веб-серверів, щоб виявити потенційні загрози, такі як шкідливе програмне забезпечення, зловмисні атаки та аномалії в поведінці користувачів. Наприклад, можна написати Python-скрипт для пошуку в журналах веб-серверів певних шаблонів, які можуть свідчити про наявність шкідливого програмного забезпечення або зловмисної атаки.

2. Сканування мережі

Python можна використовувати для сканування мережі на предмет вразливостей та шкідливого програмного забезпечення. Наприклад, можна написати Python-скрипт для використання протоколу SNMP для отримання інформації про вразливості у мережевих пристроях. Можна також написати Python-скрипт для використання протоколу NMAP для сканування мережі на предмет відкритих портів і потенційних загроз.

3. Розслідування інцидентів

Python можна використовувати для розслідування інцидентів безпеки, таких як атаки на веб-сайти або порушення даних. Наприклад, можна написати Python-скрипт для аналізу даних мережевого моніторингу, щоб виявити аномалії, які можуть свідчити про наявність атаки. Можна також написати Python-скрипт для аналізу журналів веб-серверів, щоб отримати інформацію про те, як відбулася атака.

4. Можливість працювати з машинним навчанням

Python має потужні можливості для роботи з машинним навчанням. Це дозволяє використовувати його для розробки моделей машинного навчання, які можуть допомогти виявити загрози. Наприклад, можна використовувати машинне навчання для виявлення аномалій у мережевому трафіку або для класифікації шкідливого програмного забезпечення.

5. Можливість працювати з штучним інтелектом

Python також має можливості для роботи з штучним інтелектом. Це дозволяє використовувати його для розробки агентів штучного інтелекту, які можуть допомогти виявити загрози. Наприклад, можна використовувати штучний інтелект для аналізу журналів веб-серверів або для розслідування інцидентів безпеки.

5. Можливість працювати з хмарної інфраструктури

Python має можливості для роботи з хмарної інфраструктури. Це дозволяє використовувати його для виявлення загроз у хмарних середовищах. Наприклад, можна використовувати Python для аналізу журналів хмарних серверів або для сканування хмарних мереж на предмет вразливостей.

РОЗДІЛ 1: АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Безпека в мережі.

Безпека комп'ютерної мережі охоплює заходи, що вживаються будь-яким підприємством чи організацією для захисту своєї мережі та інформації за допомогою апаратних і програмних засобів. Мета полягає в гарантуванні конфіденційності та доступності даних і мережі. Кожна компанія чи організація, що опрацьовує значну кількість даних, повинна приймати рішення для протидії різноманітним кіберзагрозам.

Одним з основних аспектів безпеки мережі є захист паролем, який визначає сам користувач мережі. Останнім часом мережева безпека стала ключовою темою в області кібербезпеки, оскільки багато організацій шукають спеціалістів із відповідними навичками. Засоби безпеки мережі спрямовані на захист від різних вразливостей комп'ютерних систем, таких як:

- Користувачі
- Розташування
- Дані
- Пристрої
- Додатки

Основним принципом мережевої безпеки є захист величезних збережених даних і мереж на рівнях, які забезпечують дотримання правил і норм, які необхідно визнати перед виконанням будь-яких дій з даними.

Ці рівні:

1. Безпека фізичної мережі
2. Технічна безпека мережі
3. Адміністративна безпека мережі

Захист інформації та даних клієнтів: Мережева безпека гарантує надійний доступ та захист інформації від кіберзагроз, що є важливим для забезпечення конфіденційності даних клієнтів.

Захист від великих втрат: Мережева безпека виключає можливість серйозних втрат, які можуть виникнути через втрату даних або інші загрози безпеки, що може негативно позначитися на фінансовому стані організації.

Підтримка репутації: Загалом, мережева безпека сприяє збереженню репутації організації, оскільки вона гарантує захист конфіденційних даних та інших важливих елементів.

Таблиця 1.1 Аспекти безпеки мережі

Аспекти безпеки мережі	Характеристики
1. Фізична безпека мережі	● Захист від неавторизованого персоналу ● Захист конфіденційності мережі від несанкціонованого доступу ● Використання зовнішніх периферійних пристроїв та маршрутизаторів для кабельних з'єднань ● Можливість використання біометричних систем
2. Технічна безпека мережі	● Захист збережених даних в мережі ● Захист даних, що переходять через мережу ● Забезпечення цілісності інформації ● Захист від зловмисних атак
3. Аміністративний захист мережі	● Контроль за поведінкою користувачів ● Авторизація та процес надання дозволу ● Управління рівнем складності для захисту від усіх атак

Мережева безпека є галуззю кібербезпеки, яка фокусується на захисті комп'ютерних мереж від кіберзагроз. Основні завдання безпеки мережі включають у себе тривале запобігання несанкціонованому доступу до мережевих ресурсів, виявлення та припинення поточних кібератак і порушень безпеки, а також забезпечення того, щоб авторизовані користувачі мали безпечний доступ до необхідних мережевих ресурсів у момент їх необхідності.

Системи безпеки мережі працюють на двох рівнях: на периметрі та всередині мережі.

На периметрі засоби безпеки намагаються зупинити кіберзагрози від проникнення в мережу. Але мережеві зловмисники іноді проникають, тому групи IT-безпеки також контролюють ресурси всередині мережі, як-от ноутбуки та дані. Навіть якщо нападники проникнуть, вони не матимуть свободи. Ця стратегія — розподіл кількох засобів контролю між хакерами та потенційними вразливими місцями — називається «глибиним захистом».

1.2 Фаєрволи: надійний бар'єр безпеки в сучасних мережах

Фаєрволи є першим бар'єром захисту, який стоїть між внутрішньою мережею користувача і небезпечним Інтернетом. Вони фільтрують трафік на основі заздалегідь визначених правил, що дозволяє чи блокує певні види комунікації. Це ефективно обмежує доступ для небажаних підключень та захищає мережу від потенційно шкідливих загроз.

Фаєрволи можуть бути апаратними (вбудованими у мережеве обладнання) або програмними (запускаються на окремих серверах чи спеціалізованих пристроях). Апаратні фаєрволи часто використовуються на рівні маршрутизаторів чи комутаторів, тоді як програмні рішення можуть надавати більше гнучкості та конфігураційних можливостей.

Основний принцип роботи фаєрволів - це застосування правил фільтрації до мережевого трафіку. Ці правила визначають, який трафік повинен бути допущений або заборонений в залежності від його характеристик, таких як IP-адреса, порт, протокол та інші атрибути. Фаєрволи також використовують станові таблиці для визначення статусу з'єднань, що дозволяє взяти до уваги попередні обміни даними між вузлами.

Фаєрволи інтегруються з системами виявлення та запобігання вторгненням (IDS/IPS), щоб виявляти та блокувати підозрілу активність. Це допомагає вчасно виявляти потенційні загрози та запобігати вторгненням в мережу.

Фаєрволи часто використовуються для підтримки віртуальних приватних мереж (VPN), які забезпечують безпеку та конфіденційність при обміні даними через неприватні мережі, такі як Інтернет. Вони також можуть використовуватися для шифрування мережевого трафіку, забезпечуючи додатковий рівень безпеки.

Фаєрволи втілюють сучасні стандарти безпеки та є ключовим інструментом для будь-якої організації, яка прагне захистити свої мережеві ресурси від різноманітних загроз. З їх допомогою можна створити надійний

бар'єр безпеки, що виконує критичну роль у збереженні цілісності та конфіденційності мережевих даних.

1.3 Контроль доступу до мережі (NAC)

Рішення для контролю доступу до мережі (NAC) можна порівняти з воротарями, які проводять процедури автентифікації та авторизації користувачів для визначення, хто може отримати доступ до мережі та які дії їм дозволено виконувати всередині неї. Термін "автентифікація" вказує на перевірку того, чи є користувач тим, за кого себе видає. З іншого боку, "авторизація" передбачає видання дозволу автентифікованим користувачам для використання мережевих ресурсів.

Часто рішення NAC використовуються для застосування політик управління доступом на основі ролей (RBAC), де права користувачів залежать від їхніх посадових обов'язків. Наприклад, молодший розробник може редагувати код, але не публікувати його, в той час як старший розробник може виконувати всі ці дії. RBAC сприяє запобіганню витоку даних, обмежуючи доступ неавторизованих користувачів до ресурсів, на які вони не мають права.

Крім того, окрім автентифікації користувачів, деякі рішення NAC можуть проводити оцінку ризиків для кінцевих точок користувачів з метою запобігання доступу до мережі незахищеним або скомпрометованим пристроям. Якщо користувач намагається підключитися до мережі з пристроєм, де застосоване застаріле антивірусне програмне забезпечення чи неправильні конфігурації, то рішення NAC блокує доступ. Деякі розширені інструменти NAC можуть навіть автоматично виправляти невідповідності в параметрах кінцевих точок.

Контроль доступу до мережі (NAC), також відомий як контроль доступу до мережі, — це процес обмеження доступу неавторизованих користувачів і пристроїв до корпоративної чи приватної мережі. NAC гарантує, що лише

аутентифіковані користувачі та пристрої, авторизовані та сумісні з політикою безпеки, можуть увійти в мережу.

У міру того, як кінцеві точки поширюються в організації (як правило, керується політикою використання власного пристрою (BYOD) і розширенням використання пристроїв Інтернету речей (IoT), потрібен більший контроль. Навіть найбільші IT-організації не мають ресурсів, щоб вручну налаштувати всі використовувані пристрої. Автоматизовані функції рішення NAC є значною перевагою, скорочуючи час і пов'язані з цим витрати на автентифікацію та авторизацію користувачів і визначення сумісності їхніх пристроїв.

Крім того, кіберзлочинці добре знають про збільшення використання кінцевих точок і продовжують розробляти та запускати складні кампанії, які використовують будь-які вразливості в корпоративних мережах. З більшою кількістю кінцевих точок поверхня атаки збільшується, що означає більше можливостей для шахраїв отримати доступ. Рішення NAC можна налаштувати для виявлення будь-якої незвичної або підозрілої активності в мережі та негайного реагування, наприклад ізоляції пристрою від мережі, щоб запобігти потенційному поширенню атаки.

Незважаючи на те, що IoT і BYOD змінили рішення NAC, NAC також служить постійною інвентаризацією користувачів, пристроїв і їх рівня доступу. Він служить активним інструментом виявлення для виявлення раніше невідомих пристроїв, які могли отримати доступ до всієї мережі або її частини, вимагаючи від IT-адміністраторів коригування політики безпеки.

Крім того, організації можуть вибрати, як NAC буде автентифікувати користувачів, які намагаються отримати доступ до мережі. IT-адміністратори можуть вибрати багатофакторну автентифікацію (MFA), яка забезпечує додатковий рівень безпеки для комбінацій імені користувача та пароля.

Обмеження доступу до мережі також означає контроль над додатками та даними в мережі, які зазвичай є метою кіберзлочинців. Чим сильніший контроль мережі, тим важче будь-якій кібератаці буде проникнути в мережу.

Які переваги контролю доступу до мережі?

1. Контроль доступу до мережі має ряд переваг для організацій:
2. Контролювати користувачів, які входять в корпоративну мережу
3. Контролюйте доступ до додатків і ресурсів, до яких користувачі хочуть отримати доступ
4. Дозвольте підрядникам, партнерам і гостям входити в мережу за потреби, але обмежте їхній доступ
5. Розділіть співробітників на групи на основі їх посадових функцій і створіть політики доступу на основі ролей
6. Захистіть себе від кібератак, встановивши системи та елементи керування, які виявляють незвичайну або підозрілу активність
7. Автоматизація реагування на інциденти
8. Створюйте звіти та статистичні дані про спроби доступу в усій організації

1.4 Системи виявлення та запобігання вторгненням (IDPS)

Систему виявлення та запобігання вторгненням (IDPS) — яку іноді називають системою запобігання вторгненням (IPS) — можна розгорнути безпосередньо за брандмауером для сканування вхідного трафіку на наявність загроз безпеці. Ці інструменти безпеки розвинулися на основі систем виявлення вторгнень (IDS) , які лише позначали підозрілу активність для перевірки. IDPS мають додаткову можливість автоматично реагувати на можливі порушення, наприклад, блокуючи трафік або скидаючи з’єднання. IDPS особливо ефективні у виявленні та блокуванні атак грубої сили та атак на відмову в обслуговуванні (DoS) або розподілених атак на відмову в обслуговуванні (DDoS) .

Система виявлення та запобігання вторгнень у мережу (IDPS) являє собою перетин між виявленням і запобіганням вторгненням. Він відстежує та аналізує мережевий трафік на наявність підозрілої чи ненормальної активності. Виявлення потенційної загрози базується на поєднанні глибокого навчання та попередньо визначених правил.

Іноді для цього може знадобитися надсилання сповіщення адміністратору мережі. IDPS може блокувати трафік з адреси джерела або навіть залучати інші рішення кібербезпеки, щоб пом’якшити потенційну атаку. Деякі рішення IDPS можуть навіть нейтралізувати атаки, видаляючи зловмисне програмне забезпечення або код.

Як і більшість злочинців, загрозливі особи найкраще працюють, коли вони можуть діяти в тіні, подалі від цікавих очей команд безпеки. Дозволити їм це зробити – не варіант. Ваше завдання — пролити світло на ваших нападників і зупинити їх на шляху.

Системи виявлення вторгнень (IDS) і системи запобігання вторгненням (IPS) разом дозволяють робити саме це.

Типи виявлення та запобігання вторгненням:

Окрім виявлення та запобігання вторгненню в мережу, існує кілька додаткових типів технології IDPS.

Таблиця 1.2 додаткові типи технології IDPS

Бездротовий	Бездротові системи IDP призначені для аналізу бездротових мереж і мережевих протоколів. Хоча вони мають ту саму основну мету, що й мережевий IDPS, вони, як правило, не аналізують вищі мережеві протоколи. Це пояснюється тим, що, на відміну від NIDPS, WIDPS не

	бачить весь трафік у мережі — скоріше, він працює шляхом безперервної вибірки мережевого трафіку.
На основі хоста	IDPS на основі хосту розгортається як агент на одному пристрої, хості або кінцевій точці — як правило, це критично важливо для бізнесу або дуже конфіденційно. Характеристики, які контролюються цим типом рішення, включають системні журнали, запущені процеси, мережевий трафік, доступ до файлів і модифікацію, а також зміни конфігурації.
Аналіз поведінки мережі	У той час як мережевий IDPS спеціально перевіряє трафік у точках доступу до мережі, система аналізу поведінки мережі перевіряє мережу, використовуючи штучний інтелект для виявлення та позначення ненормальної поведінки.

Типи виявлення та запобігання вторгненням

Як випливає з назви, є дві сторони функціональності IDP — виявлення та запобігання. Або, іншими словами, як система виявляє та усуває загрози. Відповідно до Національного інституту стандартів і технологій існує три основні класи методологій виявлення .

1. Виявлення на основі сигнатур базується на зіставленні дій або моделей поведінки з відомими загрозами. Це може включати відомі характеристики зловмисного програмного забезпечення, нестандартні конфігурації системи або явні порушення політики безпеки організації. Основним недоліком виявлення на основі сигнатур є те, що воно не здатне зрозуміти чи оцінити складні комунікації чи розпізнати раніше невідому загрозу.

2. Виявлення на основі аномалій порівнює мережеву активність із встановленим базовим рівнем, позначаючи будь-які відхилення як потенційно шкідливі. Єдиним реальним недоліком виявлення аномалій є те, що для навчання системи та створення профілю поведінки для кожного об’єкта в мережі потрібен час. Недосвідчений користувач також може ненавмисно включити шкідливу діяльність як частину профілю.

3. Аналіз протоколу з підтримкою стану подібний до виявлення аномалій, за винятком того, що він використовує розроблені постачальником профілі поведінки, які можна застосовувати універсально. Хоча вони можуть бути ресурсомісткими, вони часто визначають атаки, які інші методи пропускають. Зважаючи на це, можна обдурити систему аналізу протоколів із збереженням стану, маскуючи зловмисну активність прийнятною поведінкою протоколу.

Профілактичні можливості можуть бути як пасивними, так і активними і включають:

- Завершення сеансу TPC
- Активація вбудованого брандмауера
- Регулювання використання пропускної здатності
- Зміна або видалення шкідливого вмісту
- Переналаштування пристроїв безпеки мережі
- Активація стороннього скрипта або програми

Контроль доступу до мережі (Network Access Control, NAC) - це стратегічний підхід до управління та регулювання доступу користувачів, пристроїв і систем до комп'ютерної мережі. Основна мета NAC полягає в

забезпеченні безпеки мережі, контролі над пристроями і зменшенні ризиків з точки зору безпеки.

1.5 Віртуальні приватні мережі (VPN)

Віртуальна приватна мережа (VPN) захищає особистість користувача, шифруючи його дані та маскуючи його IP-адресу та місцезнаходження. Коли хтось використовує VPN, він підключається не безпосередньо до Інтернету, а до захищеного сервера, який підключається до Інтернету від його імені.

VPN можуть допомогти віддаленим працівникам безпечно отримувати доступ до корпоративних мереж, навіть через незахищені публічні з'єднання Wi-Fi, такі як ті, які є в кафе та аеропортах. VPN шифрують трафік користувача, захищаючи його від хакерів, які можуть захотіти перехопити їхні повідомлення.

Замість VPN деякі організації використовують доступ до мережі з нульовою довірою (ZTNA). Замість використання проксі-сервера ZTNA використовує політики контролю доступу з нульовою довірою для безпечного підключення віддалених користувачів. Коли віддалені користувачі входять у мережу через ZTNA, вони не отримують доступу до всієї мережі. Натомість вони отримують доступ лише до певних активів, які їм дозволено використовувати, і їх потрібно повторно перевіряти щоразу, коли вони отримують доступ до нового ресурсу. Дивіться розділ «Підхід нульової довіри до безпеки мережі» нижче, щоб детальніше дізнатися, як працює безпека нульової довіри.

VPN працюють шляхом маршрутизації підключення пристрою до Інтернету через спеціально налаштовану мережу віддаленого сервера, якою керує служба VPN. Це означає, що всі дані, які передаються через VPN-з'єднання, не лише зашифровані. Він маскується за віртуальною IP-адресою, що дозволяє приховати вашу особу та місцезнаходження.

Замість того, щоб надсилати ваш інтернет-трафік (пошуки, відвідані сторінки, завантаження, завантаження) безпосередньо до вашого постачальника послуг Інтернету (ISP), VPN спочатку направляє ваш трафік через сервер VPN. Таким чином, коли ваші дані нарешті передаються в Інтернет, здається, що вони надходять із сервера VPN, а не з вашого особистого пристрою.

Без VPN вашу IP-адресу — спеціальний номер, унікальний для вашої мережі — видно в Інтернеті. VPN маскує вашу IP-адресу , діючи як посередник і перенаправляючи ваш трафік. Він також додає шифрування або тунель навколо вашої особи під час підключення. Комбінація VPN-сервера та тунелю шифрування блокує вашого Інтернет-провайдера, уряди, хакерів та будь-кого іншого від шпигування за вами під час навігації в Інтернеті.

Яка у вас IP-адреса? Правильний тип може змінити ситуацію, коли йдеться про безпеку в Інтернеті. Дізнайтеся про тонкощі вашої IP-адреси за допомогою наших посібників із загальнодоступних і приватних IP-адрес і статичних і динамічних IP-адрес .

VPN працюють на рівні операційної системи, тому вони перенаправляють увесь ваш трафік через інші сервери. Це означає, що весь ваш онлайн-трафік, а також ваше фізичне місцезнаходження залишаються прихованими, поки ви переглядаєте веб-сторінки. Коли ви отримуєте доступ до сайту через сервер VPN, джерелом вашого з’єднання відображається один із багатьох VPN-маршрутизаторів, який називається проксі-сервером , а не ваш власний. Тож власники сайту та будь-хто інший, хто намагається шпигувати за вами, не можуть зрозуміти, хто ви.

Захист VPN — це найкраще, що можна наблизити до справжньої анонімності в Інтернеті, не використовуючи мережу Tor , яка пропускає ваше з’єднання навколо широко розповсюдженої мережі добровольчих ретрансляторів, фактично зберігаючи вашу веб-активність у постійному русі, щоб ніхто не міг на ній зосередитися. VPN не використовують цей (дуже повільний) протокол, але вони пропонують достатній — і суттєвий — захист,

коли ви подорожуєте сучасними дерегульованими та потенційно переповненими хакерами кібермагістралями.

Коли справа доходить до рішень конфіденційності в Інтернеті, VPN, Tor і веб-проксі – це всі варіанти, але VPN пропонує найкращий баланс комплексної безпеки та швидкості .

Незалежно від того, чи бажаєте ви залишатися в безпеці в загальнодоступній мережі Wi-Fi , приховати своє місцезнаходження від постачальників вмісту та рекламодавців або захистити себе від крадіжки особистих даних, VPN збереже вашу конфіденційність. Він пропонує потужний захист, особливо в поєднанні з іншими інструментами кібербезпеки, як-от безпечний додаток для обміну повідомленнями .

Підозрюєте, що ви стали жертвою крадіжки особистих даних? Дуже важливо діяти швидко. Дізнайтеся, як повідомити про крадіжку особистих даних , щоб зменшити збитки, потім видаліть свою особисту інформацію в Інтернеті та вживіть заходів, щоб запобігти крадіжці особистих даних .

Користуватися VPN безпечно, якщо його надає надійний постачальник VPN і використовується правильно. Надійні постачальники послуг VPN відстоюють конфіденційність із прозорою політикою конфіденційності для своїх користувачів. Небезпечний провайдер VPN може таємно відстежувати та реєструвати вашу онлайн-діяльність, мати вразливі місця в безпеці, що призводять до витоку даних, або навіть продавати ваші дані рекламодавцям.

Але хоча VPN захищає конфіденційність, безпеку й анонімність вашого інтернет-з’єднання, ви все одно ризикуєте отримати зловмисне програмне забезпечення та фішингові атаки . Отже, хоча це може зменшити ваш ризик злому та онлайн-трекерів, навіть найкращі VPN не замінять найвищий рейтинг антивірусу .

Використання VPN є законним у більшості країн. Законність використання служби VPN також залежить від країни та її геополітичних відносин з іншою

країною. Надійний і безпечний VPN завжди є законним, якщо ви не збираєтеся використовувати його для будь-яких незаконних дій, як-от шахрайство в Інтернеті, кіберкрадіжка або, у деяких країнах, завантаження захищеного авторським правом вмісту.

Згідно з повідомленням Bloomberg, Китай вирішив заблокувати всі VPN (віртуальні приватні мережі) до наступного року. Багато китайських користувачів Інтернету використовують VPN для приватного доступу до веб-сайтів, які заблоковані китайським так званим «великим брандмауером». Це робиться для того, щоб уникнути будь-якого витоку інформації в країни-конкуренти та посилити інформаційну безпеку.

VPN також мають обмежене використання для корпоративних організацій. Інфраструктура VPN не підтримує модель кібербезпеки з нульовою довірою, що важливо для організацій, які покладаються на хмарні програми та віддалених працівників.

Сьогодні доступ до надійних хмарних програм і даних є важливим для продуктивності. Доступ до мережі з нульовим довірою (ZTNA) працює як довірений брокер, який забезпечує зв’язок між користувачами та програмами. ZTNA ізолює ресурси від Інтернету та захищає їх від зовнішніх загроз.

Замість VPN корпоративні організації повинні розглянути програмно-визначений периметр (SDP). Замість того, щоб зосереджуватися на захисті мережі, SDP працює над захистом користувачів, програм і з’єднань між ними за допомогою:

Надання доступу лише після автентифікації користувача та пристрою

Призначення кожному пристрою унікального ідентифікатора для детального доступу до даних

Оскільки SDP вимагає чітких дозволів доступу та ідентифікації пристрою, він генерує детальні журнали аудиту, які є корисними для реагування на інциденти та аналізу

1.6 Безпека програми

Безпека додатків означає кроки, які команди безпеки вживають для захисту додатків і інтерфейсів програмування додатків (API) від мережевих зловмисників. Оскільки сьогодні багато компаній використовують програми для виконання ключових бізнес-функцій або обробки конфіденційних даних, програми є звичайною мішенню для кіберзлочинців. А оскільки так багато бізнес-додатків розміщено в загальнодоступних хмарах, хакери можуть використовувати їх уразливості, щоб зламати мережі приватних компаній.

Заходи безпеки програми захищають програми від зловмисників. До поширених інструментів захисту програм належать брандмауери веб-програм (WAF), самозахист програми під час виконання (RASP), статичне тестування безпеки програми (SAST) і динамічне тестування безпеки програми (DAST).

Безпека програми — це важливий аспект розробки та експлуатації програмного забезпечення. Забезпечення безпеки додатків та API допомагає уникнути ризиків і забезпечує захист від потенційних загроз. Ось деякі ключові поняття та інструменти, які використовуються для забезпечення безпеки програм:

Брандмауери веб-програм (WAF): Це системи безпеки, розташовані між веб-додатком та Інтернетом. Вони фільтрують та моніторять HTTP-трафік між веб-додатком і користувачами, блокуючи атаки типу SQL-ін'єкцій, кросс-сайтового скриптінгу (XSS) та інших вразливостей.

Самозахист програми під час виконання (RASP): RASP-технології аналізують та моніторять активний код програми під час виконання. Якщо вони виявляють аномалії чи підозрілу активність, вони можуть автоматично реагувати, блокуючи потенційні загрози та запобігаючи вразливостям.

Статичне тестування безпеки програми (SAST): Цей підхід використовується для аналізу вихідних кодів програм під кутом зору безпеки, знаходячи потенційні вразливості та помилки ще до етапу виконання програми.

Динамічне тестування безпеки програми (DAST): DAST використовується для тестування безпеки програми в реальних умовах виконання. Він взаємодіє з програмою як зовнішній користувач, шукаючи вразливості під час роботи додатку. Коректне управління ідентифікацією та автентифікацією:

- Забезпечення безпеки здебільшого розпочинається з ефективного управління ідентифікацією користувачів і забезпечення безпеки процесу автентифікації.
- Застосування шифрування для захисту конфіденційної інформації, особливо під час передачі через мережі.
- Регулярні оновлення програмного забезпечення та виправлення вразливостей дозволяють уникати використання вже відомих загроз.
- Системи моніторингу, які слідкують за подіями безпеки в реальному часі, дозволяють виявляти та реагувати на потенційні загрози швидше.
- Забезпечення того, щоб персонал розумів основи кібербезпеки та дотримувався кращих практик у цьому напрямку.

1.7 Безпека електронної пошти

Інструменти безпеки електронної пошти можуть допомогти запобігти фішинговим атакам та іншим спробам скомпрометувати облікові записи електронної пошти користувачів. Більшість служб електронної пошти мають

вбудовані засоби безпеки, такі як фільтри спаму та шифрування повідомлень. Деякі інструменти безпеки електронної пошти містять пісочниці, ізольовані середовища, де групи безпеки можуть перевіряти вкладення електронної пошти на наявність зловмисного програмного забезпечення , не відкриваючи мережу.

В програмах для обробки електронної пошти, миттєвого листування та шифрування файлів, таких як PGP, використовуються різні методи шифрування для забезпечення безпеки повідомлень. Наприклад, у поштових клієнтах, таких як Microsoft Outlook та Mozilla Thunderbird, можна встановити шифрування повідомлень за допомогою асиметричних алгоритмів, вимагаючи реєстрації ключа користувача.

Програми миттєвого листування, такі як Telegram, Viber та WhatsApp, використовують гібридне шифрування end-to-end, що дозволяє забезпечити конфіденційність повідомлень. Вони використовують симетричний алгоритм шифрування при передачі повідомлення та обміні ключами за допомогою протоколу Діффі-Геллмана.

Програма PGP використовує симетричні алгоритми для шифрування даних, а ключі, створені для них, шифруються асиметричними алгоритмами. Це забезпечує криптографічну стійкість та ефективність роботи алгоритму. Важливою перевагою цього методу є безпечне передавання шифрованого ключа.

Помітно, що всі ці програми забезпечують цілісність та конфіденційність повідомлень. У поштових клієнтах додається цифровий підпис для захисту від порушень цілісності та ідентифікації автора листа. Усі використовувані алгоритми мають високу стійкість до криптоаналізу, що робить їх ефективними засобами забезпечення безпеки електронної переписки.

Безпека електронної пошти є вирішальним компонентом для захисту як бізнесу, так і особистих активів від загроз. Електронна пошта широко відома як вектор загроз номер один для кібератак, і кіберзлочинці постійно змінюють свою тактику та методи, щоб використовувати вразливості електронної пошти.

Відповідно до звіту Verizon про розслідування витоку даних, 94% шкідливих програм було доставлено електронною поштою. Згідно з даними звіту Cisco, 96% усіх фішингових атак відбуваються через електронну пошту.

Ефективна безпека електронної пошти передбачає більше, ніж просто використання технологій для виявлення загроз і захисту даних і цифрових активів. Це також навчання персоналу щодо безпеки електронної пошти та розуміння складної природи сучасних загроз. Глобальний звіт Terranova Phishing Benchmark Global Report показав, що 67,5% людей, які натиснули фішингове посилання, швидше за все, надішлють свої облікові дані на пов'язаному фішинговому веб-сайті.

Впровадження заходів безпеки електронної пошти допомагає захистити конфіденційну інформацію від потрапляння в чужі руки та гарантує, що ваша організація продовжує відповідати нормам захисту даних, таким як GDPR та HIPAA. Захищена система електронної пошти також може запобігти дорогим простоям, спричиненим зловмисними електронними листами, які скомпрометували вашу мережеву інфраструктуру.

У цифровому середовищі, що постійно розвивається, де спілкування електронною поштою залишається фундаментальним для повсякденних операцій, важко переоцінити важливість надійної стратегії безпеки електронної пошти для забезпечення безперебійної роботи вашого бізнесу та захисту від потенційних загроз.

1.8 Висновки до розділу

У даному розділі ми детально розглянули різні аспекти безпеки в мережі, що є надзвичайно важливими в сучасному цифровому світі. Відомості,

представлені в цьому розділі, демонструють різноманітність і складність вимог до забезпечення безпеки в інформаційних системах та мережах.

Перше, що було розглянуто, це загальна концепція безпеки в мережі. Мережі стають все більш важливими для нашого повсякденного життя, і вони стають потенційними місцями атак та порушень безпеки. Важливо забезпечити ці мережі надійними заходами безпеки.

Далі, було розглянуто фаєрволи, системи, які відповідають за контроль трафіку в мережі і захищають її від небажаних підключень та атак. Вони є важливими компонентами забезпечення безпеки мережі.

Контроль доступу до мережі (NAC) та системи виявлення та запобігання вторгненням (IDPS) є іншими важливими елементами безпеки мережі. Вони дозволяють ідентифікувати та відстежувати незвautorизований доступ і втручання в мережу, а також приймати відповідні заходи для їх запобігання.

Віртуальні приватні мережі (VPN) важливі для забезпечення конфіденційності та захисту даних, передаваних через неприватні мережі, такі як Інтернет.

Безпека програм та електронної пошти також важливі, оскільки багато атак спрямовані на служби програм та електронну пошту.

Загалом, цей розділ наголошує на важливості розуміння та впровадження різноманітних засобів та стратегій для забезпечення безпеки в мережах та інформаційних системах. Правильне використання цих інструментів допоможе у збереженні конфіденційності, цілісності та доступності даних та ресурсів мережі, що є надзвичайно важливим у сучасному цифровому світі.

РОЗДІЛ 2: ІНСТРУМЕНТИ ДЛЯ ВИЯВЛЕННЯ ЗАГРОЗ

2.1 Огляд мови програмування Python в контексті кібербезпеки

Мова програмування Python в останні роки набула значної популярності в галузі кібербезпеки. Це пов'язано з кількома факторами, які роблять Python відмінним вибором для розробки безпечних і ефективних інструментів та додатків.

Python відомий своєю простотою та легкістю в освоєнні. Читабельний синтаксис допомагає розробникам швидко розуміти та аналізувати код, що є важливим у кібербезпеці для виявлення та усунення потенційних уразливостей.

Python має велику та активну спільноту розробників, яка постійно вносить внески в розвиток мови. Це означає, що існує велика кількість бібліотек, фреймворків та інструментів, які спрощують розробку безпечних програм та забезпечують широкі можливості для кіберзахисту.

Python має різноманітні бібліотеки та фреймворки для роботи з криптографією, мережевою безпекою, обробкою даних та іншими аспектами кібербезпеки. Наприклад, бібліотека cryptography дозволяє легко використовувати криптографічні функції.

Багато інструментів для тестування безпеки, таких як Metasploit, Wireshark, Burp Suite, використовують Python для реалізації своєї функціональності. Це дозволяє розробникам та експертам з кібербезпеки швидко адаптувати ці інструменти та розширювати їхні можливості.

Python дозволяє розробникам створювати швидкі прототипи та експериментувати з ідеями швидше, що є важливим у сфері кібербезпеки, де швидкість реакції на нові загрози грає ключову роль.

Таблиця 2.1 порівняння бібліотек Python

Бібліотека	Опис	Основні можливості	Ліцензія	Спільнота/Популярність
Scapy	Мережевий інструмент для маніпуляцій пакетами	Створення, відправка, захоплення, аналіз пакетів	GPLv2	Велика
PyCrypto	Бібліотека криптографічних функцій	Шифрування, хешування, підписи, генерація ключів	Custom License	Середня
Cryptography	Сучасна криптографічна бібліотека	Шифрування, сертифікати, безпечне зберігання ключів	Apache License 2.0	Велика
Yara	Інструмент для ідентифікації і класифікації зразків	Правила для виявлення малварі, аналіз файлів	BSD-3-Clause	Середня
Requests	Легкий HTTP клієнт	Відправка HTTP/HTTPS запитів, взаємодія з веб-сервісами	Apache License 2.0	Висока
Beautiful Soup	Бібліотека для парсингу HTML та XML	Збір даних з веб-сторінок, веб-скрапінг	MIT License	Висока
Python Nmap	Інтерфейс до Nmap	Сканування мережі, виявлення вразливостей	GPLv2	Середня

Python є переносимою мовою програмування, що означає, що програми, написані на Python, можуть працювати на різних операційних системах та архітектурах, що спрощує розгортання та використання кіберзахисних рішень.

Незважаючи на всі переваги, важливо також враховувати, що правильна реалізація та конфігурація безпечних застосунків важлива незалежно від мови програмування. Python може бути потужним інструментом для розробки безпечних програм, але без відповідного розуміння принципів кібербезпеки, використання мови само по собі не гарантує безпеку програмного забезпечення.

Python — це надзвичайно функціональна та потужна мова програмування, яку можна використовувати для широкого спектру проектів, особливо коли йдеться про безпеку.

1) Виявлення вразливостей

Власники бізнесу повинні знати про потенційні ризики, створені зловмисниками. На щастя, Python спрощує та прискорює процес пошуку та усунення вразливостей у системі.

Він надає надійні бібліотеки для перевірки мережевого трафіку на наявність підозрілих дій і може бути використаний для створення програм, які шукають слабкі місця у вихідному коді. Python дозволяє фахівцям із безпеки захищати свої системи від пошкоджень, швидко виявляючи та усуваючи будь-які поточні чи майбутні недоліки.

2) Безпека веб-додатків

Оскільки безпека веб-додатків стає все більш важливою для компаній, мова програмування Python надає розробникам потужні ресурси для безпечного кодування. Такі бібліотеки, як Django та Flask, захищають від таких загроз, як впровадження SQL, міжсайтове виконання сценаріїв і віддалене виконання коду.

За допомогою Requests, Scrapy та Urllib розробникам легко отримувати інформацію з веб-сторінок. Нарешті, pyJWT і криптографія забезпечують безпечні засоби аутентифікації користувачів і шифрування даних. Використання

цих інструментів і бібліотек Python може допомогти розробникам створювати безпечні веб-додатки.

3) Криптографія

Криптографія є важливим елементом кібербезпеки, а Python є однією з найкращих мов програмування для побудови криптосистем. Криптографія захищає дані, кодуючи їх таким чином, що їх неможливо прочитати без певного ключа.

Python полегшує це завдяки широкому спектру бібліотек, які надають доступ до різноманітних алгоритмів, протоколів і утиліт для створення безпечних систем шифрування. Крім того, Python можна використовувати для створення унікальних криптографічних рішень, таких як алгоритми хешування, симетричні та асиметричні алгоритми та алгоритми цифрового підпису. Це також корисно для налаштування систем автентифікації з двофакторною автентифікацією, переконавшись, що конфіденційну інформацію можуть переглядати лише ті, хто має дозвіл.

Python широко використовується для таких додатків, як технологія блокчейн, які для безпеки значною мірою залежать від криптографії. Технологія блокчейн стає все більш популярною в різних сферах через її неперевершений захист і надійність. Таким чином, цілком природно, що Python є основною мовою для розробників, які прагнуть створювати захищені блокчейн-програми.

Крім того, Python є дуже корисним інструментом для розробників, які працюють над безпечними криптографічними рішеннями. Його універсальність і широка бібліотечна допомога роблять його оптимальною мовою для створення безпечних криптографічних рішень. Крім того, не дивно, що Python швидко стає улюбленою мовою професіоналів з кібербезпеки, завдяки можливості швидкої розробки та впровадження рішень.

4) Автоматизація завдань

Використання автоматизації в сфері кібербезпеки є надзвичайно вигідним. Ця автоматизація може значно підвищити ефективність, а також дозволити вам проводити регулярні та точні заходи безпеки. Python пропонує швидкий і легкий спосіб використовувати автоматизацію для всіх ваших завдань безпеки. Веб-розробники використовуючи Python для створення сценаріїв, які сканують уразливості у вашій мережі, перевіряють комп'ютери, на яких було встановлено виправлення, виявляють вторгнення та навіть розміщують приманки для зловмисників. Усі ці сценарії можна налаштувати на запуск за розкладом, тож вам не потрібно вручну контролювати їх. Крім того, Python можна використовувати для спрощення адміністративних завдань, таких як керування обліковими записами користувачів і налаштування мережевих пристроїв. Користувальницькі сценарії, створені за допомогою Python, можуть навіть допомогти у тестуванні безпеки, створюючи тестові випадки та запускаючи автоматичне сканування.

Автоматизація є чудовим інструментом для полегшення та захисту вашої мережі. Використовуючи Python, ви можете створити сценарії, які візьмуть на себе завдання безпеки та заощадять значну кількість часу.

5) Машинне навчання

Машинне навчання є безцінним активом для сфери кібербезпеки. Ця технологія дозволяє організаціям завчасно виявляти зловмисну активність і запобігати загрозам до їх виникнення. Вивчаючи шаблони в даних, можна запрограмувати алгоритми, щоб передбачати майбутні події та їхні наслідки. У світі кібербезпеки машинне навчання регулярно використовується для виявлення зловмисного програмного забезпечення, фішингових атак та інших потенційно шкідливих дій. Завдяки своїй здатності оцінювати величезні обсяги даних і виявляти будь-які особливості, він служить корисним механізмом безпеки. Крім того, компанії можуть отримати вигоду від технологій машинного навчання, оскільки вони можуть розробити більш точні правила безпеки та автоматизувати оцінку мережевого трафіку.

З огляду на його надзвичайні можливості, використання машинного навчання у сфері кібербезпеки зросло в геометричній прогресії за останні роки, забезпечуючи додатковий рівень захисту від кібератак і потенціал для організацій для перехоплення небезпеки на її ранніх стадіях.

6) Злом паролів

Захист бізнес-систем від потенційних атак є важливим завданням, яке передбачає аналіз і виявлення слабких місць у паролях. Python є універсальним інструментом для злому паролів, оскільки він надає багато корисних бібліотек і модулів для бізнесу. Наприклад, 'passlib', бібліотека Python, може обчислити ентропію та визначити, чи достатньо надійний пароль. Ентропія вимірює рівень випадковості, який є вирішальним для створення безпечних паролів. Ця бібліотека також надає функції для створення хешів і солей, додатково захищаючи паролі. Крім того, підприємства можуть використовувати «Джон Різник», безкоштовний інструмент злому паролів, щоб виявити слабкі паролі, що зберігаються в онлайнових і офлайнових базах даних. Крім того, Python дозволяє власникам бізнесу створювати спеціальні сценарії для автоматизації процесу злому паролів, заощаджуючи час і зусилля. Розробники програмного забезпечення для створення власних налаштувань кібербезпеки у програмному забезпеченні. Python є незамінним інструментом для власників бізнесу, коли мова заходить про виявлення та виправлення слабких місць у паролях. Його бібліотеки та модулі можуть допомогти власникам бізнесу виконувати різноманітні завдання, наприклад створювати хеші, генерувати солі та автоматизувати завдання.

7) Криміналістика

Криміналістика відіграє важливу роль у кібербезпеці, оскільки дозволяє слідчим досліджувати цифрові докази у випадках кіберзлочинності. Python можна використовувати для проведення різноманітних криміналістичних операцій, таких як пошук шаблонів, відновлення артефактів і більш ефективна оцінка доказів. Python також цінний у створенні звітів, які слугуватимуть

доказом у суді. Автоматизація процесу криміналістики за допомогою Python допомагає пришвидшити процес і надає точні результати, заощаджуючи час і ресурси.

8) Зворотне проектування

Реверсивне проектування – це процес аналізу, який дозволяє нам ідентифікувати компоненти, особливості та структуру системи. У контексті кібербезпеки він використовується для вивчення шкідливих програм, щоб отримати розуміння того, як вони працюють і що потрібно зробити, щоб їх викорінити. Це може допомогти виявити вихідний код шкідливого програмного забезпечення, виявити приховані елементи та отримати уявлення про те, як діють кібер-зловмисники.

Python надає кілька корисних бібліотек, які дозволяють розробникам розбивати двійкові файли, отримувати дані та виявляти шкідливий код. Крім того, вони дозволяють розробникам імітувати код і відстежувати шлях даних. Цей тип оцінки можна використовувати для перевірки зловмисного програмного забезпечення, допомагаючи визначити його походження, призначення та інші важливі деталі.

Скориставшись можливостями зворотного проектування Python, спеціалісти з кібербезпеки можуть отримати глибоке розуміння зловмисних програм і бути більш підготовленими проти потенційних ризиків. Крім того, це дозволяє їм створювати стратегії для швидкої та ефективної боротьби з кіберзагрозами.

9) Безпека мережі

Оскільки компанії переводять свої системи в хмару, вони повинні забезпечити захист своїх даних. За допомогою Python інженери мережевої безпеки можуть розгортати автоматизацію для різних процесів, які можуть бути тривалими та трудомісткими.

Python можна використовувати для створення сценаріїв для виявлення несанкціонованого доступу, постійного спостереження за мережею та

розпізнавання загроз. Його також можна використовувати для зміцнення мережевої інфраструктури, такої як маршрутизатори та комутатори, з автоматизованими завданнями налаштування.

Також можна створювати приманки, залучаючи зловмисників для відстеження їхніх переміщень.

Python корисний як для внутрішньої, так і для зовнішньої безпеки. Наприклад, його можна використовувати для систем автентифікації та авторизації, формування безпечних паролів і шифрування конфіденційної інформації.

Використання Python для мережевої безпеки забезпечить організаціям підвищений рівень безпеки, зменшивши їхній ризик кібератак. Завдяки різноманітним використанням Python є оптимальним вибором для компаній, які прагнуть випереджати потенційні загрози.

10) Безпека IoT

Інтернет речей (IoT) стрімко стає частиною нашого життя. Але з усіма підключеними пристроями виникають потенційні ризики для безпеки. Python може допомогти з безпекою IoT, надаючи можливість виявляти, аналізувати та пом’якшувати атаки на під’єднані пристрої.

Python можна використовувати для аналізу мережевого трафіку на наявність шкідливих дій, виявлення вразливих пристроїв і контролю доступу до них. Завдяки можливостям машинного навчання Python його також можна використовувати для виявлення та блокування зловмисного коду на пристрої IoT.

Ця мова програмування також може допомогти з процесами автентифікації та авторизації підключених пристроїв, надаючи доступ лише авторизованим користувачам. Крім того, Python можна використовувати для шифрування зв’язку між пристроями, гарантуючи безпеку будь-яких конфіденційних даних, які надсилаються через мережу.

Розробників мобільних додатків , переконайтеся, що вони знають, як реалізувати IoT за допомогою Python. Python можна використовувати для створення автоматизованих процесів для забезпечення регулярного оновлення та оновлення пристроїв IoT. Це допомагає захистити пристрої, зменшуючи ймовірність успішного використання експлойтів.

Він також може стати потужним інструментом для підвищення безпеки IoT у 2023 році. Він має можливості виявляти й аналізувати загрози, а також захищати зв'язок і процеси оновлення. Оскільки все більше й більше нашого життя стає пов'язаним через Інтернет речей, важливо, щоб ми могли захистити ці пристрої від атак.

11) Тестування на проникнення

Тестування на проникнення є ключовим компонентом кібербезпеки, який оцінює безпеку системи, програми або мережі шляхом пошуку вразливостей. Python є надійним інструментом для проведення такої форми тестування, що дозволяє користувачам автоматизувати завдання, обробляти результати та шукати нові вразливості.

Найпопулярнішою бібліотекою Python для тестування на проникнення є Paramiko, яка дозволяє користувачам отримувати доступ до віддалених систем за допомогою SSH. Він також надає доступ до тунелювання, переадресації портів і підтримки проксі-сервера. Іншими корисними бібліотеками Python для тестування на проникнення є Scapy і Snort, перша забезпечує маніпуляції з пакетами, а друга забезпечує систему виявлення вторгнень у мережу.

Сценарії Python можна використовувати для сканування мереж і систем на наявність слабких місць, а також для автоматизації таких завдань, як злам ненадійних паролів або сканування вразливостей. Звіти з детальною інформацією про будь-які вразливості або недоліки, які були виявлені під час тестування, також можна створити за допомогою Python.

Це гарантує швидке виявлення та вирішення будь-яких проблем, запобігаючи використанню зловмисниками системи компанії та захищаючи дані від зловмисних дій.

Використання Python для тестування на проникнення є вигідним для власників бізнесу, оскільки це не тільки допомагає заощадити час і гроші, але й гарантує безпеку їхніх систем.

2.2 Використання Python для моніторингу мережевого трафіку

Python - це потужний і універсальний мову програмування, який можна використовувати для різних завдань, включаючи моніторинг мережевого трафіку. Для цього існує кілька бібліотек та інструментів, які дозволяють збирати, аналізувати та відображати дані про мережевий трафік.

Однією з найпопулярніших бібліотек для моніторингу мережевого трафіку в Python є Scapy. Scapy дозволяє збирати пакети з мережі, розбирати їх на складові частини та аналізувати. Бібліотека також містить широкий спектр функцій для фільтрації та сортування даних, а також для генерування нових пакетів.

Ще одна популярна бібліотека для моніторингу мережевого трафіку в Python - Pcapy. Pcapy є інтерфейсом для аналізу пакетів, який використовує драйвери pcap для перехоплення пакетів з мережі. Бібліотека дозволяє збирати пакети з різних джерел, включаючи фізичні мережеві інтерфейси, віртуальні мережі та файлові потоки.

Використання Python для моніторингу мережевого трафіку передбачає застосування різноманітних бібліотек та інструментів для збору, аналізу та візуалізації даних про мережевий трафік. Основною ідеєю є використання різноманітних бібліотек, таких як Scapy, pycap, або socket, для захоплення та аналізу мережевого трафіку.

Аналіз мережевих пакетів дозволяє отримувати інформацію про адреси, протоколи та дані, які передаються. Збирання статистики мережевого трафіку включає в себе вимірювання обсягу переданих та отриманих даних, кількості пакетів, а також визначення типів протоколів.

Для візуалізації даних про мережевий трафік можна використовувати бібліотеки для графіків та діаграм, такі як Matplotlib чи Plotly. Це дозволяє наглядно відображати зміни у мережевому трафіку та сприяє зрозумінню обсягу та характеристик комунікацій.

Додатково, можна використовувати готові інструменти для моніторингу мережі, такі як Wireshark, або розробляти власні скрипти для автоматичного збору даних та аналізу трафіку. Автоматизація цих процесів дозволяє ефективно виявляти аномалії та сповіщати про них адміністраторів системи. Такий підхід допомагає забезпечити ефективний моніторинг та безпеку мережевого середовища.

Важливим етапом є також реалізація моніторингу в реальному часі, що дозволяє оперативно реагувати на зміни у мережевому трафіку. Автоматизовані засоби сповіщення дозволяють інформувати адміністраторів про виявлення підозрілих або небезпечних ситуацій. Крім того, програми для моніторингу мережевого трафіку на Python можуть включати функціонал для зберігання та архівування статистики. Це може бути корисно для подальшого аналізу та виявлення тенденцій у розвитку мережі.

Загалом, використання Python для моніторингу мережевого трафіку дозволяє створити потужні та гнучкі інструменти, які відповідають конкретним потребам системи. Цей підхід сприяє не лише ефективному виявленню можливих проблем, а й оптимізації мережевого середовища в цілому.

Ще однією важливою аспектом використання Python для моніторингу мережевого трафіку є можливість інтеграції з іншими інструментами та системами. Наприклад, ви можете розробити зв'язок із системами журналювання

подій або іншими інфраструктурними інструментами для ще більш ефективного контролю та аналізу мережі.

Також слід враховувати аспекти безпеки під час моніторингу мережевого трафіку. Забезпечення конфіденційності та цілісності даних під час їх аналізу та зберігання є важливим завданням. Python надає інструменти для реалізації заходів безпеки та шифрування, які допомагають забезпечити відповідність з вимогами безпеки інформації.

Загалом, використання Python для моніторингу мережевого трафіку стає необхідним елементом у сучасних IT-системах. Це дозволяє не лише відслідковувати та реагувати на події в реальному часі, але й забезпечує аналіз та оптимізацію мережевого середовища для підтримки стабільної та безпечної роботи систем.

У контексті використання Python для моніторингу мережевого трафіку важливо також звертати увагу на розширені можливості обробки даних. Використання бібліотек машинного навчання, таких як TensorFlow або scikit-learn, може допомогти виявляти аномальний трафік або прогнозувати можливі проблеми на основі історичних даних.

Крім того, можна реалізувати функції збору та аналізу метрик продуктивності мережі, таких як пропускна здатність, затримки чи втрати пакетів. Це дозволяє вчасно виявляти проблеми та оптимізувати мережеві ресурси. У світі хмарних технологій і мікросервісної архітектури важливим є також моніторинг віртуалізованих середовищ та контейнерів. Python може використовуватися для розробки інструментів, які автоматизують збір та аналіз даних з різних рівнів інфраструктури.

Необхідно також враховувати можливості масштабування та розширюваності ваших інструментів моніторингу. Розробка модульних рішень та використання архітектурних патернів дозволить легко додавати новий функціонал та адаптувати систему під зростання обсягу мережевого трафіку чи зміну вимог до моніторингу.

2.2 Розробка, вибір інструментів для виявлення аномального поведінки в мережі з використанням Python

Машинне навчання (ML) прагне зрозуміти поведінку змінних, виявляючи шаблони в даних. Дослідники, які займаються кількісним аналізом, використовують ці закономірності та розробляють моделі для прогнозування. На жаль для розробників, шаблони (або тренди) можуть зламатися. Розрив шаблону — це щось незвичайне — аномалія. Щоб виявити аномалії в даних, ви використовуєте процес ML під назвою виявлення аномалій.

Викид — це значення у випадковій вибірці або колекції спостережень, яке аномально далеко від інших значень. Відстань, яка класифікується як ненормальна, залежить від набору даних і варіанту використання. Не всі викиди є аномаліями, але щоб визначити аномалії в наборі даних, вам потрібно знайти викиди.

Викиди діляться на три великі категорії:

Глобальні або точкові викиди : незвичайна подія, яка різко відрізняється від усіх інших точок даних. Наприклад, уявіть модель ML, навчену прогнозувати ціну будинку на основі таких факторів, як розмір і місце розташування. Глобальний викид тут може бути будинком, який значно більший або менший за інші будинки в наборі даних, або розташований у незвичному місці порівняно з іншими будинками.

Контекстуальні або умовні викиди : незвичайна подія в конкретному контексті. Розглянемо модель ML, навчену передбачати ймовірність переходу клієнта на продукт конкурента на основі віку, доходу чи інших факторів. У цьому випадку контекстний викид може бути молодим клієнтом із високим доходом, який значно відрізняється від інших молодих клієнтів у наборі даних. Цей викид не може бути незвичайним, якщо розглядати його в контексті всього набору

даних, але він буде незвичайним, якщо розглядати лише підмножину молодих клієнтів.

Сукупні викиди : унікальна ситуація, коли одна аномалія не викликає тривоги, а сукупність таких аномалій. Уявіть собі модель ML, навчену прогнозувати успіх маркетингової кампанії на основі таких факторів, як демографічні дані цільової аудиторії та тип кампанії. Тут сукупний викид може являти собою групу сегментів аудиторії зі схожими демографічними показниками, на які спрямована кампанія одного типу, але зі значно нижчим рівнем відповіді, ніж інші порівняні сегменти аудиторії.

Ви можете використовувати алгоритми виявлення аномалій, щоб запобігти найгіршим сценаріям, швидко й точно визначаючи незвичні точки даних або шаблони в наборі даних. Існують різні алгоритми виявлення аномалій, зокрема контрольовані, неконтрольовані та напівконтрольовані.

Контрольовані алгоритми навчаються з використанням позначених даних і можуть точно ідентифікувати аномальні точки даних на основі їх відхилень від нормальних даних.

Неконтрольовані алгоритми використовують статистичні методи для виявлення незвичайних точок даних на основі їх відхилення від загального розподілу даних.

Напівконтрольовані алгоритми поєднують контрольовані та неконтрольовані елементи навчання та навчаються з використанням суміші позначених і немаркованих даних.

Requirements:

Python 3.10.8

NumPy 1.22

Pandas 1.5 або вище

Matplotlib 3 або вище

Seaborn 0,12 або вище

Scikit-learn 1.1 або вище

Зі зростанням залежності від комп'ютерних мереж для різноманітних операцій важливість мережевої безпеки стала надзвичайно високою. Організації в різних галузях постійно стикаються з проблемою захисту своїх мереж від потенційних загроз, таких як зараження зловмисним програмним забезпеченням, несанкціонований доступ і витік даних. Традиційні засоби безпеки, такі як брандмауери та системи виявлення вторгнень, можуть ефективно протистояти відомим загрозам. Однак ці традиційні підходи часто не можуть ідентифікувати нові та еволюційні загрози.

Щоб вирішити цю проблему, в останні роки значну увагу приділяють методам машинного навчання. Алгоритми машинного навчання можуть аналізувати великі обсяги даних мережевого трафіку та виявляти аномальні шаблони, які можуть вказувати на потенційні загрози безпеці. Використовуючи історичні дані та навчаючись на минулих інцидентах, ці алгоритми можуть ідентифікувати відхилення від нормальної поведінки мережі, забезпечуючи раннє виявлення загроз і можливість проактивних дій.

Основною метою використання машинного навчання для виявлення аномалій мережевого трафіку є розробка моделей, які можуть точно класифікувати мережевий трафік як звичайний або зловмисний. Це передбачає використання навчальних моделей на позначених наборах даних мережевого трафіку та різноманітних функцій, таких як розмір пакета, тип протоколу, джерело/адресат IP-адреси та номери портів. Потім ці моделі можна розгортати в режимі реального часу для постійного моніторингу мережевого трафіку і створення сповіщень при виявленні аномальної поведінки.

Переваги використання машинного навчання для виявлення аномалій мережевого трафіку численні. Це дозволяє організаціям виявляти загрози, які традиційні заходи безпеки можуть пропустити, надаючи додатковий рівень безпеки. Крім того, машинне навчання може адаптуватися та вчитися на нових даних, підвищуючи свою точність з часом для боротьби з еволюційними загрозами. Це також допомагає мінімізувати помилкові спрацьовування,

зменшуючи кількість непотрібних сповіщень і дозволяючи командам безпеки ефективніше зосереджуватися на справжніх загрозах.

Технології машинного навчання стали потужним інструментом для виявлення аномалій мережевого трафіку та можливості раннього виявлення загроз. Аналізуючи великі обсяги даних мережевого трафіку та виявляючи аномальні шаблони, ці алгоритми можуть підвищити безпеку мережі та зменшити потенційні загрози. Впровадження стратегій машинного навчання в сферу мережевої безпеки дозволяє організаціям бути на крок попереду зловмисників та зберігати свої цінні цифрові активи.

Чандола та інші досліджують мережеві аномалії як відхилення від звичайної або відомої поведінки в мережі, які, за їхньою думкою, мають наслідки для безпеки. Ці аномалії також відомі як нетипові дії, спрямовані на порушення нормальної роботи мережі. Аномалії визначаються закономірностями в даних, які не відповідають чітко визначеній концепції логічного стану (13).

Zhao та інші вказують, що нераціональні та значущі аномалії в механізмі передачі можна розглядати як аномалії (24).

Ahmad та інші (2017) визначають миттєву стадію, на якій відбуваються дії системи, значно відхиляючись від її попередньої нормальної поведінки (16).

Мохд Алі (2018) зауважує, що різні автори використовують різні терміни, такі як аномалії, викиди або винятки, щоб посилатися на аномалії мережі, що призводить до плутанини в термінології. Для розуміння концепції аномалій в мережевій системі важливо зрозуміти, що вважається нормальним. Існують три основних типи мережевих аномалій: точкові, контекстуальні та колективні (6).

Ансамблеві методи, відомі також як системи з кількома класифікаторами, передбачають навчання моделей за допомогою кількох методів машинного навчання для створення комбінованих класифікаторів, що підвищує точність (Абуромман та ін., 2017). Деякі системи виявлення вторгнень (IDS) були розроблені на основі принципу механізмів кластеризації, як зазначено в літературі.

Одна з таких систем, розроблена Гу та ін. (2019), є IDS SVM, що використовує метод опорних векторів і включає функцію збільшення. Вони застосували співвідношення інтенсивності до базових кривих, що покращило якість навчання даних. Експериментальні результати показали, що група SVM досягла конкурентоспроможних результатів з точки зору точності спостережень у моніторингу, швидкості навчання, ясності та обговорення неправдивих новин даних порівняно з іншими методами. Для оцінки використовувався набір даних NSL-KDD.

Фам та ін. (2018) надають дані в рамках певної структури та механізму для отримання переваг, спрямованих на покращення продуктивності IDS. Вони використовували механізми кластеризації з механізмами на основі дерева як основні дані. Робота показала, що використання даних J48 поліпшило точність класифікації та зменшило частоту помилкових тривог у наборах даних NSL-KDD. У роботі Bhatti et al. (2020) для спостереження за даними розроблено алгоритм на основі класифікації варіантів нападу. Границя виконання складалася з чотирьох основних етапів: укладання даних, попередня обробка, процедура навчання та прийняття рішень. Окремі класифікатори навчалися окремо, а їхні рішення об'єднувалися більшістю голосів.

Зазвичай, в машинному навчанні використовуються інтелектуальні алгоритми на етапі моделювання, які поділяються на чотири основні типи навчання: контрольоване, неконтрольоване, напівконтрольоване та навчання з підкріпленням.

1. Контрольоване навчання:

Контрольоване навчання розділяється на два основні класи: класифікація та регресія. Класифікація визначає дані у попередньо визначені категорії, надаючи мітку для кожної категорії. Наприклад, визначення, чи належить екземпляр даних до категорії "човен" чи "автомобіль". Регресія використовується для прогнозування безперервних значень, таких як час або вага. Деякі алгоритми для контрольованого навчання включають дерева рішень,

випадкові ліси, найближчі сусіди (KNN), багатошаровий перцептрон (MLP), машини опорних векторів (SVM), наївний Баєс, лінійна або логістична регресія та інші.

2. Неконтрольоване навчання:

У неконтрольованому навчанні вхідні дані не мають попередньо визначених вихідних міток, і немає контролю для визначення закономірностей. Цей підхід використовується для виявлення шаблонів в доступних даних без чіткого спрямування від людини.

А. Асоціація:

Ця техніка в основному спрямована на пошук найпоширеніших наборів елементів у великих обсягах даних. Вона визначає зв'язки між змінними ознак та вказує на ефективні змінні для майбутніх значень. Прикладом є аналіз ринку, де правила асоціації можуть виявити, які елементи часто купуються разом.

Б. Кластеризація:

Цей метод використовується для розділення даних на групи, які не мають попередньо визначених властивостей та визначаються як кластери. Кластер - це група екземплярів даних, що описує природну структуру даних. Елементи в одному кластері схожі один на одного, і вони відрізняються від елементів в інших кластерах. Деякі елементи можуть не входити в жоден кластер, і їх називають викидами.

3. Напівконтрольоване навчання:

У напівконтрольованому навчанні як позначені, так і непозначені вхідні дані використовуються для тренування моделі, причому кількість непозначених даних значно переважає кількість позначених. Це поєднання підсилює ефективність та точність навчання. Таким чином, напівконтрольоване навчання є комбінацією контрольованого та неконтрольованого навчання.

2.3 Автоматизація аналізу безпеки за допомогою Python

Процес машинного навчання може бути розділений на сім етапів, як описано на малюнку. Перший етап - це збір даних, який включає в себе виявлення, збільшення та генерацію даних. Для дослідження нових наборів даних важливо використовувати доступні загальнодоступні набори через Інтернет чи дослідницькі установи. Після виявлення даних, збільшення відбувається шляхом підсилення поточних баз даних за рахунок додавання зовнішніх даних. У випадках, коли наявний набір даних недостатній і зовнішні дані недоступні, може бути використана генерація даних, навіть як альтернативний підхід .

Після отримання набору даних важливо піддати його передобробці, щоб забезпечити його відповідність вхідним вимогам моделі. Цей процес, відомий як підготовка даних або попередня обробка, забезпечує перетворення необроблених даних в формат, придатний для використання моделлю.

На наступному етапі вибирається відповідна модель для ефективного вирішення конкретної проблеми. Важливо враховувати, що складна модель не завжди гарантує кращі результати, тому вибір моделі має спрямовуватися на досягнення навчання на наборі даних.

Наступний етап - це навчання моделі за допомогою позначених даних. Процес включає поступове вдосконалення прогнозів моделі, зміну упереджень та ваг під час кожної ітерації. У той час як контрольоване машинне навчання використовує позначені дані, неконтрольоване машинне навчання використовує непозначені дані.

Після завершення навчання починається етап оцінки моделі, де тести проводяться на частині даних, яка раніше не використовувалася для навчання. Це визначає успіх моделі в реальних умовах. Після оцінки модель може бути вдосконалена, оптимізуючи параметри та збільшуючи кількість ітерацій.

В кінцевому результаті, коли всі етапи виконано, можна використовувати модель для знаходження рішення реальних проблем.

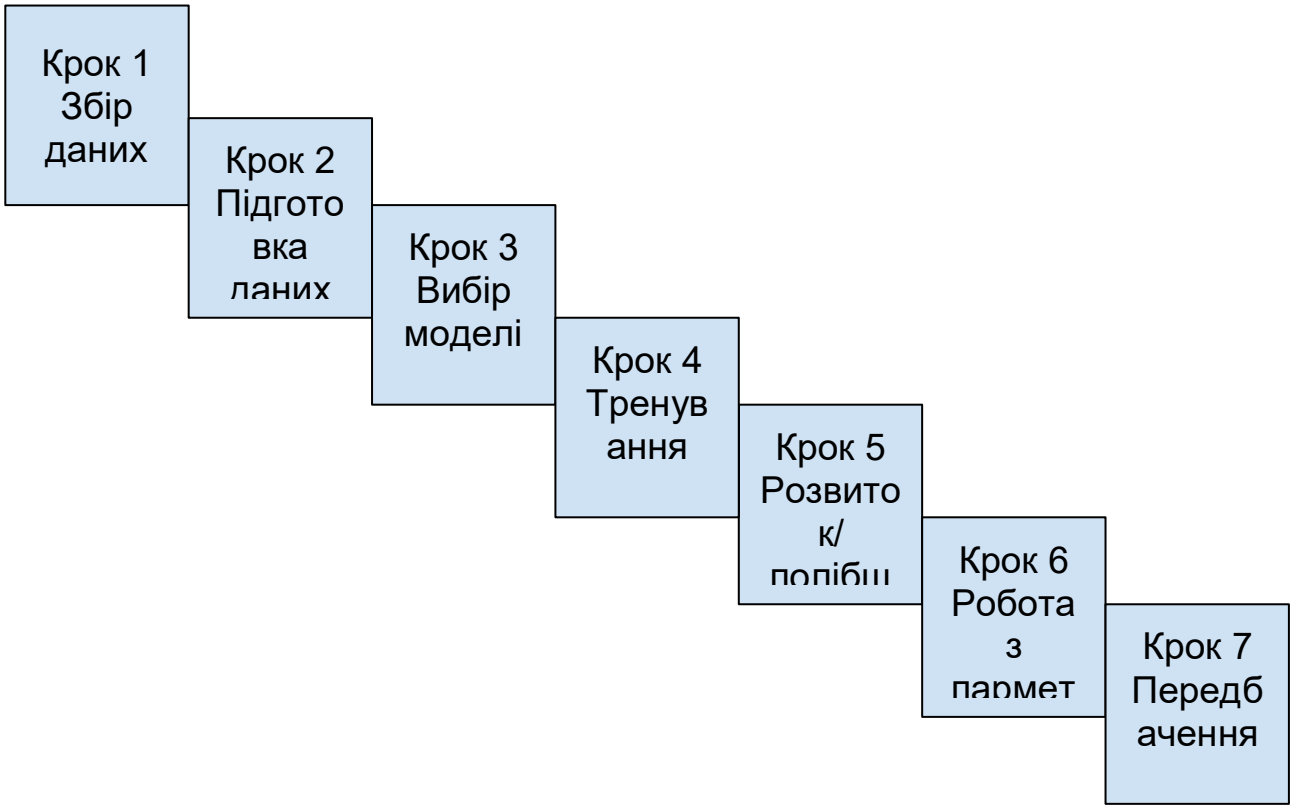

Рис. 1.1. 7 кроків машиного навчання

2.5 Інтеграція Python із системами виявлення та запобігання вторгненням

Інтеграція Python у системи виявлення та запобігання вторгненням (IDS/IPS) відкриває нові можливості для підвищення безпеки мережі. Python може використовуватися для розробки скриптів, які аналізують мережевий трафік, розпізнають підозрілу поведінку, автоматизують відповіді на загрози та навіть інтегруються з іншими системами безпеки. Ось декілька прикладів, як можна використовувати Python у цьому контексті:

Аналіз трафіку: Python може використовуватися для написання скриптів, які аналізують мережевий трафік, виявляючи аномалії, що можуть вказувати на спробу вторгнення.

Розробка правил IDS/IPS: Python може бути використаний для автоматизації процесу створення та оновлення правил виявлення, що дозволяє швидко реагувати на нові загрози.

Інтеграція з іншими системами безпеки: Python може використовуватися для інтеграції IDS/IPS з іншими інструментами безпеки, такими як системи управління інформацією та подіями безпеки (SIEM) та іншими аналітичними інструментами.

Автоматизація відповідей на інциденти: Python може використовуватися для написання скриптів, які автоматизують відповіді на виявлені інциденти, наприклад, шляхом автоматичного блокування IP-адрес або ізоляції вразливих систем.

Машинне навчання та ШІ: Використання Python для розробки моделей машинного навчання, що допомагають виявляти складні шаблони в мережевому трафіку, які можуть вказувати на вторгнення або інші кіберзагрози.

Кастомізація засобів моніторингу: Розробка власних інструментів моніторингу та аналізу, які точно відповідають специфічним потребам організації.

Python є гнучким і потужним інструментом, який може значно підвищити ефективність систем виявлення та запобігання вторгненням, забезпечуючи кращу захищеність мережевої інфраструктури.

Python може бути виключно корисним для підсилення систем виявлення та запобігання вторгненням (IDS/IPS). Через свою гнучкість і потужність, Python дозволяє автоматизувати та оптимізувати багато аспектів роботи з IDS/IPS.

Аналіз мережевого трафіку є однією з ключових областей, де Python виявляє свої переваги. Бібліотеки як Scapy дозволяють маніпулювати та

аналізувати мережеві пакети, що надає можливість глибоко зануритися в деталі мережевих операцій. PyShark, що працює як інтерфейс до Wireshark, може використовуватися для більш високорівневого аналізу трафіку.

Автоматизація створення правил IDS/IPS є ще однією важливою областю застосування Python. Це може включати генерацію або оновлення правил на основі аналізу загроз або реакцію на нові відомі загрози. Цей підхід значно підвищує швидкість та гнучкість реагування на нові виклики безпеки.

Завдяки бібліотекам для машинного навчання, як-то scikit-learn, TensorFlow та Keras, Python стає потужним інструментом у виявленні аномалій і розробці алгоритмів, які можуть виявляти складні шаблони поведінки в мережевому трафіку, що можуть вказувати на кібератаки.

Окрім того, Python може бути використаний для інтеграції IDS/IPS з іншими системами безпеки, такими як SIEM (Системи Управління Інформацією та Подіями Безпеки). Це дозволяє централізувати збір та аналіз даних про безпеку, забезпечуючи більш широкий огляд стану мережевої безпеки.

У сукупності, Python є інструментом, що дозволяє значно підвищити ефективність та реактивність систем IDS/IPS, допомагаючи в розробці більш гнучких та адаптивних рішень у сфері кібербезпеки.

2.5 Висновки до розділу

У цьому розділі ми розглянули різні аспекти використання мови програмування Python у контексті кібербезпеки. Python став популярним інструментом серед професіоналів у сфері кібербезпеки завдяки своїй простоті використання, широкій підтримці бібліотек та інструментів, а також спроможностям для автоматизації та розробки різноманітних рішень.

В першому підрозділі ми зробили огляд мови Python та її основних можливостей в контексті кібербезпеки. Python має велику спільноту розробників, що активно працюють над створенням бібліотек та інструментів, які спрощують роботу з мережевими протоколами, обробкою даних і автоматизацією завдань в сфері безпеки.

У наступних підрозділах ми розглянули конкретні застосування Python у кібербезпеці. Ми дослідили використання Python для моніторингу мережевого трафіку, що дозволяє виявляти аномальну активність та потенційні загрози для безпеки мережі. Також була розглянута розробка та вибір інструментів для виявлення аномального поведінки в мережі за допомогою Python, що стає важливим завданням для забезпечення безпеки мереж та систем. Нарешті, ми зупинилися на автоматизації аналізу безпеки за допомогою Python, що дозволяє ефективно обробляти велику кількість даних і відслідковувати потенційні загрози.

Загальний висновок полягає в тому, що Python виявляється потужним інструментом для спеціалістів у галузі кібербезпеки. Його простота використання, розширюваність та здатність до автоматизації дозволяють створювати ефективні та надійні рішення для забезпечення безпеки мереж та інформаційних систем. Python відіграє важливу роль у виявленні загроз, моніторингу мереж, та аналізі безпеки, і його застосування в цій сфері продовжує зростати.

РОЗДІЛ 3: РОЗРОБКА МОДЕЛІ МАШИННОГО НАВЧАННЯ ДЛЯ АВТОМАТИЧНОГО ВИЯВЛЕННЯ ФІШИНГОВИХ ВЕБ-САЙТІВ.

3.1 Опис задачі

Фішинг — це кіберзлочин, який передбачає використання шахрайських електронних листів, повідомлень і веб-сайтів для викрадення конфіденційної інформації, наприклад паролів, даних кредитної картки та інших особистих даних. Із зростанням Інтернету та онлайн-транзакцій фішингові атаки стають дедалі складнішими, що ускладнює їх виявлення та уникнення.

Фішинг все ще залишається одним із найкращих і найуспішніших способів хакерів виманити з нас наші гроші та викрасти нашу особисту та фінансову інформацію.

Сучасні фішингові атаки складні, і виявити їх стає все важче. Згідно з опитуванням Intel, 97% спеціалістів із безпеки не можуть відрізнити законні електронні листи від фішингових.

Машинне навчання може бути потужним інструментом для виявлення фішингових веб-сайтів. Навчаючи алгоритми машинного навчання на великому наборі даних як законних, так і шахрайських веб-сайтів, алгоритми можуть навчитися розрізняти їх. Це може призвести до розробки ефективних систем виявлення фішингу, які можуть автоматично ідентифікувати та попереджати користувачів про потенційно небезпечні веб-сайти.

Існує кілька типів алгоритмів машинного навчання, які можна використовувати для виявлення фішингу, зокрема контрольоване навчання, неконтрольоване навчання та глибоке навчання. Алгоритми контрольованого навчання навчаються на позначених даних, де функції кожного веб-сайту використовуються для класифікації його як легітимного чи фішингового. З

іншого боку , алгоритми неконтрольованого навчання групують веб-сайти на основі їхніх особливостей, дозволяючи виявляти викиди, які можуть свідчити про фішингові веб-сайти.

Алгоритми глибокого навчання, такі як згорткові нейронні мережі (CNN) , використовують складні архітектури нейронних мереж для аналізу функцій веб-сайтів і прогнозування.

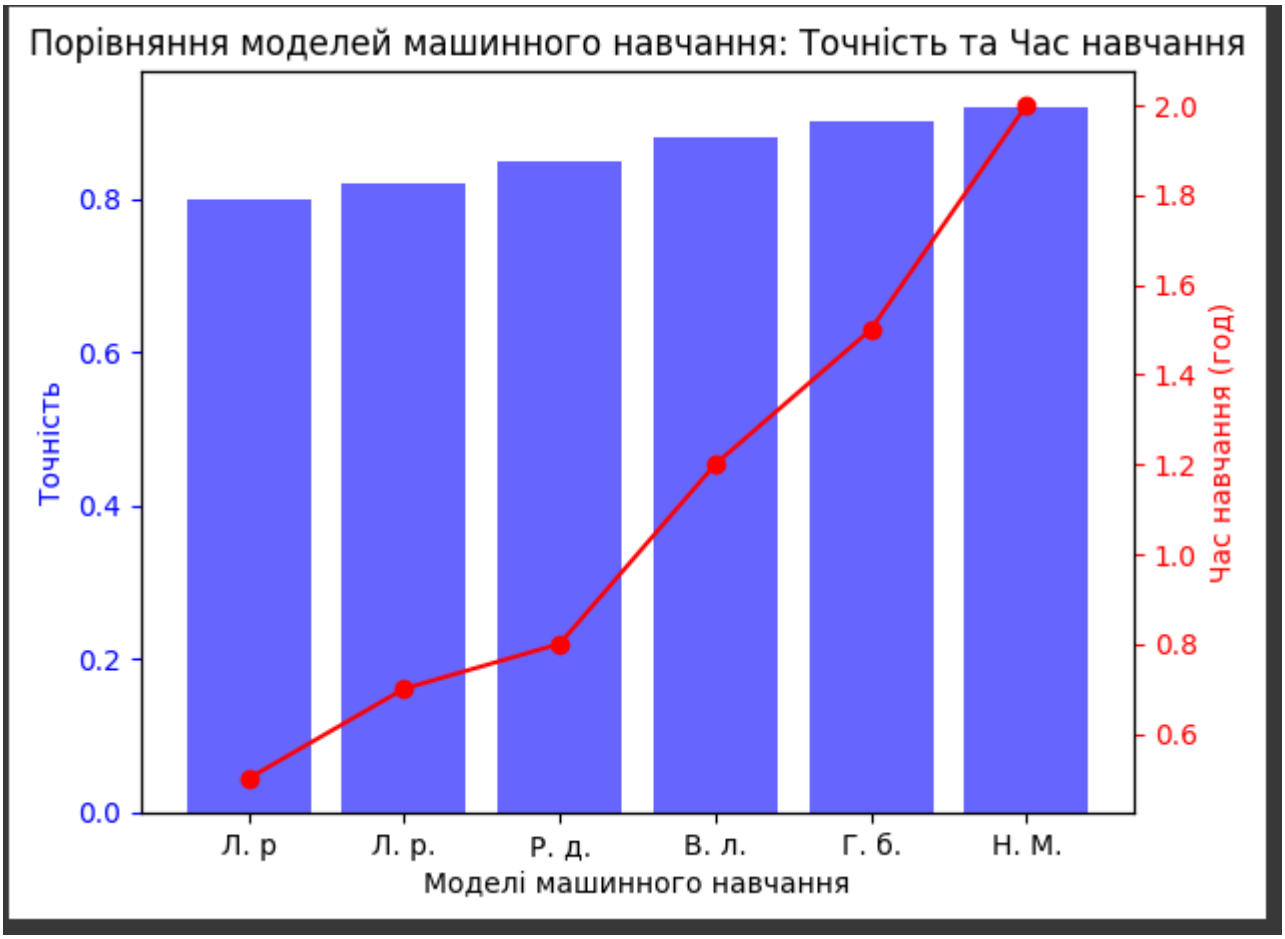

Рис. 3.2 Графік порівняння моделей

Скорочення до рис 3.1.

'Лінійна регресія', 'Логістична регресія', 'Рішення дерев', 'Випадковий ліс', 'Градієнтний бустінг', 'Нейронні мережі'

Однією з переваг використання машинного навчання для виявлення фішингу є те, що воно може бути більш точним і ефективним, ніж традиційні методи, такі як чорні списки або системи на основі евристики. Це пояснюється тим, що алгоритми машинного навчання можуть навчитися ідентифікувати

фішингові веб-сайти на основі їхніх функцій, а не покладатися на попередньо визначені правила чи підписи. Це робить їх більш надійними та менш схильними до хибнопозитивних чи хибнонегативних результатів.

Під час навчання алгоритмів машинного навчання для виявлення фішингу важливо використовувати великий і різноманітний набір даних веб-сайтів. Це допоможе гарантувати, що алгоритми зможуть вивчати та виявляти фішингові веб-сайти, які представляють різні типи фішингових атак, які існують. Крім того, слід ретельно вибирати функції, які використовуються алгоритмами для розрізнення законних веб-сайтів від фішингових. Загальні функції, які використовуються для виявлення фішингу, включають структуру URL-адреси, вміст веб-сайту та візуальні підказки, такі як використання офіційних логотипів або сертифікатів безпеки.

Ще одна перевага використання машинного навчання для виявлення фішингу полягає в тому, що його можна легко інтегрувати в існуючі системи безпеки та робочі процеси. Наприклад, алгоритми машинного навчання можна використовувати для автоматичного сканування вхідних електронних листів і позначення будь-яких повідомлень, які містять посилання на фішингові веб-сайти. Їх також можна інтегрувати в розширення браузера, дозволяючи користувачам отримувати попередження про потенційно небезпечні веб-сайти, перш ніж вони їх відвідають.

Незважаючи на численні переваги використання машинного навчання для виявлення фішингу, є деякі обмеження та проблеми, які необхідно вирішити. Одна з головних проблем полягає в тому, щоб алгоритми могли виявляти нові типи фішингових атак. Це вимагає постійного оновлення навчальних даних і функцій, які використовуються алгоритмами. Крім того, алгоритми машинного навчання можуть бути вразливими до агресивних атак , коли зловмисники маніпулюють функціями фішингових веб-сайтів, щоб уникнути виявлення. Щоб вирішити цю проблему, важливо використовувати надійні та безпечні моделі машинного навчання, стійкі до цих атак.

3.2 Реалізація моделі

У нашому методі ми використовуємо лише інформацію про URL-адресу веб-сайту, щоб визначити, чи є веб-сайт фішинговим. Таким чином, немає необхідності фактично відвідувати веб-сайт, щоб визначити, чи є він фішинговим чи ні. Це також дозволяє користувачеві не відвідувати фішингові веб-сайти та наражати себе на шкідливі коди, які вони можуть нести. Алгоритм випадкового лісу можна застосувати до набору даних, що містить такі функції, що містять метадані URL-адрес. Алгоритм випадкового лісу у нашій задачі пропонує перевагу, оскільки дані також не переобладнуються.

Ми починаємо наш проект машинного навчання з імпорту різноманітних бібліотек Python, які нам потрібні для обробки даних, візуалізації та побудови моделей машинного навчання. Ми використовуємо numpy і pandas для роботи з даними, matplotlib.pyplot та seaborn для візуалізації, а також включаємо ключові функції з sklearn для побудови та оцінки моделей. Для аналізу URL-адрес ми використовуємо спеціальні бібліотеки, такі як tldextract і ipaddress.

Ми завантажуємо набір даних, який називається "url train dataset.csv", містить дані URL-адрес та відповідні мітки для класифікації. Ми переглядаємо дані, перевіряємо типи даних і виконуємо базовий аналіз, щоб зрозуміти, з чим ми працюємо.

Ми аналізуємо розподіл міток у наборі даних і, якщо виявляємо, що він незбалансований, ми проводимо повторну вибірку, щоб вирівняти його. Ми розробляємо ряд функцій для вилучення характеристик із URL-адрес, таких як кількість крапок, наявність дефісів та довжина URL-адреси. Ці характеристики додаємо до нового фрейму даних, який ми називаємо featureSet.

Для кращого розуміння наших даних ми використовуємо seaborn і matplotlib для візуалізації характеристик, таких як довжина URL-адрес, кількість точок, підкаталогів, субдоменів та запитів. Це допомагає нам у дослідницькому аналізі даних.

Таблиця 3.1 Порівняння найпопулярніших моделей ML

Модель	Тип моделі	Призначення	Основні переваги	Основні недоліки
Лінійна регресія	Регресія	Прогнозування числових значень	Простота, інтерпретованість	Обмежена складність моделі
Логістична регресія	Класифікація	Бінарна класифікація	Легка інтерпретація, швидкість	Обмеженість у складних зв'язках
Нейронні мережі	Глибинне навчання	Різноманітні завдання	Гнучкість, точність у складних задачах	Потреба у великих даних, складність
Дерево прийняття рішень	Ансамблеві методи	Класифікація, регресія	Інтерпретованість, простота використання	Схильність до перенавчання
Випадковий ліс	Ансамблеві методи	Класифікація, регресія	Висока точність, устойчивість до перенавчання	Вища складність, час обчислень
Градієнтний бустінг	Ансамблеві методи	Класифікація, регресія	Висока точність, ефективність	Складність в налаштуванні, час обчислень
SVM (Машина опорних векторів)	Класифікація	Бінарна та багатокласова класифікація	Висока точність у складних задачах	Обмеженість у великих даних, складність

Ми готуємо наші дані для навчання моделі, розділяючи характеристики та мітки на навчальні та тестові набори. Потім ми тренуємо кілька моделей машинного навчання, таких як Decision Tree, AdaBoost, Logistic Regression, Gaussian Naive Bayes, та KNN, на нашому наборі даних. Для кожної моделі ми

проводимо навчання на навчальному наборі та оцінюємо її продуктивність на тестовому наборі.

Ми використовуємо GridSearchCV для налаштування гіперпараметрів наших моделей, щоб знайти найкращі можливі параметри. Ми також візуалізуємо криві навчання для цих моделей, щоб побачити, наскільки ефективно вони навчаються зі збільшенням обсягу даних.

Далі ми намагаємося зменшити дисперсію нашого класифікатора Decision Tree, регулюючи його параметри. Після цього ми проводимо підсумкове тестування цього класифікатора на тестових даних, виводячи на екран матрицю точності та помилок.

Ми також застосовуємо нашу модель до нового набору даних для трансферного навчання, завантажуючи "url transfer dataset.csv" і проходячи його через той самий процес вилучення характеристик. Навчена модель Decision Tree потім використовується для прогнозування міток у цьому новому наборі даних.

Набір даих: використаний набір даних містить фішингову та нефішингову URL-адресу з міткою, яка вказує, чи є відповідна URL-адреса фішинговою.

URL	Label
http://account-google-com.ngate.my/c44cca401760e0c1e12587f9b833f8e5...	1
http://www.coffeespecialties.com/...	0
http://black.pk/wp-content/2013/04/bp.postale/96306090ce26bb6aae928b9a...	1
http://atomicsoda.com/manutd...	0
http://bostoncoffeecake.com/...	0
http://www.durst.de/...	0
http://www.jobofmine.com/...	0
http://safetem.com/fcid/login.php?cmd=login_submit&id=46775597b693a54e...	1
http://elicafe.com/...	1
http://www.vgl.ucdavis.edu/informatics/strand.php...	0
http://wellcomelibrary.org/...	0
http://www.topusajobs.com/...	0
http://www.mymothersdelicacies.com/...	0
http://iw4510.freehostgate.net/MaPe8-DoXV4/2016/f78382f3060f02b5acc285...	1
http://www.audi.co.nz/...	0
http://www.marlinaquarindo.com/Amazon/ap/login/login.php...	1
http://www.cityofdreams.com.ua/tmp/pa/8ar0rvpnelq9m6mfk5nhdcoxyg8a5hzb...	1
http://www.dreamlandbbq.com/...	0
http://www.cncsaz.com/documents/tracking.php?l=_JeHFUq_VJOXK0QWHtoGYDw...	1
http://www.barrelsandbottles.co.uk/...	0
http://everydayhomecarellc.com/public/images/abonne/194497a1fd6be8408a...	1
http://www.redcafe.net/...	0
http://tabtimeline.com/Canada_MD/009/car.php...	1
http://www.cecm.sfu.ca/~loki/Papers/Numbers/...	0
http://www.cincybabiesmatter.com/wp-admin/aa/pin.htm...	1
http://guj-epd.gov.in/...	0
http://www.sugarboo.com/...	0
http://www.mancitynews.com/...	0
http://www.dessertcafeonline.com/...	0
http://www.thinkingblocks.com/ThinkingBlocks_MD/TB_MD_Main.html...	0

Рис. 3.2Готовий дата-сет

Збір даних: Проект починається зі збору набору даних URL-адрес. Ці дані можуть включати як законні, так і фішингові URL-адреси. Важливо мати збалансований набір даних, щоб забезпечити ефективне навчання моделі.

Аналіз та обробка URL-адрес: Кожна URL-адреса аналізується на наявність характеристичних ознак фішингу. Це може включати довжину URL, використання спеціальних символів, присутність підозрілих доменних імен та інші метадані.

Використання алгоритму випадкового лісу: Модель випадкового лісу обрана для класифікації URL-адрес, оскільки вона ефективна у виявленні складних шаблонів в даних. Випадковий ліс також має переваги, такі як висока точність та здатність обробляти надмірно розміщені дані.

Навчання та оцінка моделі: Після обробки даних модель навчається на основі набору даних. Оцінка моделі виконується з використанням методів перехресної перевірки та інших метрик, таких як точність, відгук та F1-оцінка.

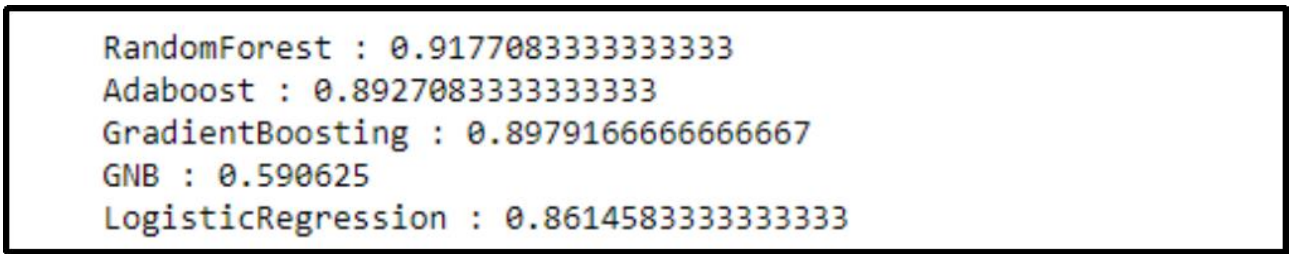

Рис. 3.3 Точність моделі

Важливо регулярно оновлювати набір даних, щоб забезпечити актуальність моделі у відповідь на нові фішингові методи. Для підвищення ефективності рекомендується інтегрувати додаткові параметри, такі як аналіз вмісту веб-сторінок або сертифікаційну інформацію.Розробка зручних інтерфейсів допоможе залучити більше користувачів та зробити технологію доступною для ширшої аудиторії.

3.3 Висновок

Підсумовуючи наш проект з машинного навчання, ми успішно реалізували повний цикл розробки, від первинної обробки даних до побудови і оцінки передових моделей класифікації. Наша головна мета була зосереджена на аналізі та класифікації URL-адрес для виявлення потенційних загроз, і ми досягли значного прогресу у цьому напрямку.

У рамках проєкту з виявлення фішингових веб-сайтів через аналіз URL-адрес було досягнуто значних успіхів у підвищенні безпеки в Інтернеті. Використання моделі випадкового лісу для класифікації URL-адрес виявилося ефективним підходом, що дозволяє користувачам уникати відвідування потенційно шкідливих веб-сайтів.

Завантаживши та аналізувавши наш початковий набір даних, ми ретельно підійшли до процесу балансування класів та інженерії характеристик, що дозволило нам отримати детальне уявлення про наші дані. Використання візуалізацій допомогло нам глибше зрозуміти розподіл та природу характеристик, які ми аналізували.

Особливо значущим було застосування нашого оптимізованого класифікатора Decision Tree на новому наборі даних для трансферного навчання. Це не тільки підтвердило гнучкість та адаптивність нашої моделі, але й продемонструвало її здатність ефективно працювати з різними наборами даних.

Функція check, яку ми розробили для демонстрації прогнозування моделі, дозволила нам практично перевірити і підтвердити наші результати, додавши додаткову впевненість у якості нашої роботи.

В цілому, цей проект не лише підсилив наше розуміння застосування машинного навчання у кібербезпеці, але й надав цінний досвід у комплексному аналізі даних, розробці характеристик, налаштуванні моделей та їх оцінці. Ми впевнені, що набуті знання та навички будуть корисними у наших майбутніх дослідженнях та проектах.

РОЗДІЛ 4: ЕКОЛОГІЧНІ ПРОБЛЕМИ УРБАНІЗАЦІЇ

4.1 Урбанізація, її причина та наслідки

Урбанізація, процес переходу людства з сільськогосподарського способу життя до міського, є одним з найбільш суттєвих та глибоких перетворень сучасного світу. За останні десятиліття міста стали центрами росту, і сьогодні більше половини світового населення проживає у міських регіонах. Цей непередбачувий розквіт міського життя призвів до ряду значущих соціальних, економічних і екологічних наслідків.

У містах люди знаходять безліч можливостей: робочі місця, освіту, доступ до культурних та розважальних заходів, а також соціальну взаємодію. Проте, зростаюча урбанізація супроводжується серйозними проблемами, які вимагають негайної уваги. Однією з найбільш важливих та нагальних з них є екологічні проблеми, пов'язані зі зростанням міських популяцій та розвитком міської інфраструктури.

Під час зростання міст та їхньої інтенсифікації, виникають складні питання щодо збереження навколишнього середовища та раціонального використання ресурсів. Екологічна стійкість та збалансованість в урбанізованому середовищі стають насущими завданнями для сучасних міст. У цій статті ми докладно розглянемо основні екологічні проблеми, які виникають внаслідок урбанізації, та визначимо можливі шляхи вирішення цих проблем у міських регіонах.

Заслуговує уваги той факт, що проблеми екології в містах несуть потенційну загрозу здоров'ю людей, біорізноманітості та стабільності клімату

нашої планети. Тому розуміння цих проблем і розробка стратегій для їхнього вирішення стають критичними завданнями для урбанізованого світу.

Урбанізація, як основний мегатренд сучасного суспільства, несе з собою комплекс екологічних проблем, що серйозно впливають на навколишнє середовище та якість життя мешканців міст. У цьому розділі ми детально розглянемо екологічні проблеми, пов'язані з урбанізацією, включаючи забруднення повітря, води та втрату біорізноманітності.

Забруднення повітря - це одна з найвідоміших та найпоширеніших проблем урбанізації. Основні аспекти цієї проблеми включають:

- Автомобільний транспорт: Зі зростанням міського населення збільшується кількість автомобілів на дорогах. Двигуни внутрішнього згоряння викидають в атмосферу оксиди азоту, вуглеводні та інші шкідливі речовини, що призводять до забруднення повітря і формування смогу.
- Промисловість: Міста часто є центрами промислового виробництва. Промислові об'єкти викидають значні обсяги шкідливих речовин, таких як діоксиди сірки та азоту, важкі метали і інші забруднюючі речовини.
- Споживання енергії: Зростання споживання електроенергії в містах призводить до викидів вуглецю та інших парникових газів. Теплова енергія, вироблена в урбанізованих районах, може також призводити до підвищення температури повітря (тепловий острів).

Забруднення води - інша серйозна екологічна проблема, пов'язана з урбанізацією. Головні аспекти цього явища включають:

- Стічні води: Зі збільшенням кількості населення та підприємств у містах зростає обсяг стічних вод, які викидаються в водойми та річки без належної очистки. Це призводить до забруднення води хімічними речовинами, бактеріями та іншими забруднюючими факторами.

- Водопостачання та водоспоживання: Зростання попиту на питну воду у містах ставить під загрозу річкові та озерні екосистеми, які постачають воду. Перекачування води з великих водойм для забезпечення міст водою також може має негативний вплив на біорізноманіття.

Втрата зелених зон та втрата біорізноманітності - це третя ключова екологічна проблема, пов'язана з урбанізацією:

- Забудова: Розширення міст призводить до втрати зелених зон, лісів та інших природних середовищ. Це не тільки призводить до втрати екологічних резерватів, але також погіршує умови для існування біорізноманітності.
- Руйнування середовища: Великі міста можуть стати бар'єрами для переміщення тварин та сприяти зменшенню популяцій деяких видів. Нафтові плями, руйнування природних біотопів та інші екологічні

4.2 Стратегії вирішення екологічних проблем

Урбанізація, незважаючи на свої численні вигоди, породжує значні екологічні виклики. Щоб забезпечити сталість міського розвитку та знизити негативний вплив на природу, необхідно впроваджувати стратегії та заходи з охорони навколишнього середовища. Нижче представлені ключові стратегії вирішення екологічних проблем урбанізації.

Перехід до використання чистих та відновлюваних джерел енергії, таких як сонячна, вітрова, та гідроенергія, може значно зменшити забруднення повітря у містах. Розвиток технологій для виробництва та зберігання енергії зеленого джерела є критично важливим для зниження емісій парникових газів та інших шкідливих речовин.

Зниження використання особистих автомобілів та покращення громадського транспорту можуть допомогти зменшити забруднення повітря в

містах. Швидкісні трамваї, метро, автобуси з низьким рівнем викидів, а також велосипедні та пішохідні інфраструктури мають стати пріоритетними вкладами у міську мобільність.

Ефективна система управління відходами, включаючи розсортування вторинних ресурсів, переробку та відновлювання, є ключовою для зменшення відходів та забруднення навколишнього середовища. Міста повинні сприяти розвитку циркулярної економіки та заохочувати управління відходами на рівні громадян.

Збереження природних зон, парків, лісів та природних біотопів у містах допомагає підтримувати біорізноманітність та покращувати якість повітря. Програми з висадки дерев, створення міських садів та підтримка місцевих природних резерватів сприяють збереженню середовища.

Освіта та інформаційна робота з громадянами важливі для формування свідомого підходу до екологічних питань. Громадянська участь у розробці та впровадженні стратегій охорони навколишнього середовища сприяє сталому розвитку міст.

Ці стратегії повинні бути інтегровані в загальний план сталого розвитку міста, який би враховував потреби мешканців та забезпечував баланс між економічним, соціальним і екологічним розвитком. Тільки спільними зусиллями можна створити міське середовище, яке було б сприятливим для життя та забезпечило б сталість екологічного розвитку урбанізованих регіонів.

4.3 Висновок до розділу

Урбанізація, незважаючи на свій неспорний розвиток та зростання економічного потенціалу, призводить до серйозних екологічних викликів. Забруднення повітря, води, втрата зелених зон і біорізноманітності стають невід'ємною частиною життя в містах. Проте існують стратегії та заходи, які можуть сприяти вирішенню цих екологічних проблем.

Перший крок - це перехід до чистих джерел енергії, що допоможе зменшити забруднення повітря та сприяти боротьбі зі зміною клімату. Покращення громадського транспорту робить міста більш доступними та зменшує тиск на дороги і атмосферу. Ефективна система управління відходами сприяє зменшенню відходів та збереженню природних ресурсів.

Збереження зелених зон і біорізноманітності є невід'ємною частиною сталого розвитку міст. Висадка дерев, створення міських садів і парків, а також збереження природних резерватів сприяють збереженню біорізноманітності і покращенню якості життя мешканців.

Найважливішим елементом є освіта та залучення громадян до участі в екологічних програмах та ініціативах. Громадяни повинні бути свідомими та активними учасниками процесу збереження навколишнього середовища та сталого розвитку міст.

У висновку можна сказати, що вирішення екологічних проблем урбанізації вимагає комплексного підходу, спільних зусиль та дієвих стратегій. Тільки завдяки цьому міста зможуть надати здорове, екологічно безпечне та стало розвинене середовище для своїх мешканців та майбутніх поколінь.

ПЕРЕЛІК ДЖЕРЕЛ ІНФОРМАЦІЇ

1. Eset. "Експлойт." [Online]. Available: https://www.eset.com/ua-ru/support/information/entsiklopediya-ugroz/eksploit/

2. GeeksforGeeks. "Види атак." [Online]. Available: https://www.geeksforgeeks.org/basic-network-attacks-in-computer-network/

3. Forcepoint. "CYBER EDU. What is Network Security?" [Online]. Available: https://www.forcepoint.com/cyber-edu/networksecurity

4. Cynet. "Network Attacks and Network Security Threats." [Online]. Available: https://www.cynet.com/network-attacks/networkattacks-and-network-security-threats/

5. GitHub - yzhao062/pyod. "A Python Toolbox for Scalable Outlier Detection (Anomaly Detection)." [Online]. Available: https://github.com/yzhao062/pyod

6. GitHub - selimfirat/pysad. "Streaming Anomaly Detection Framework in Python (Outlier Detection for Streaming Data)." [Online]. Available: https://github.com/selimfirat/pysad

7. scikit-learn. "Novelty and Outlier Detection – scikit-learn 1.0.1 documentation." [Online]. Available: https://scikit-learn.org

8. Никишова А. В. "Интеллектуальная система обнаружения атак на основе многоагентного подхода." Вестник Волгоградского государственного университета. Серия 10. Инновационная деятельность, 2011, № 5, с. 35–37.

9. Аткина В. С. "Оценка эффективности катастрофоустойчивых решений." Вестник Волгоградского государственного университета. Серия 10. Инновационная деятельность, 2012, № 6, с. 45–48.

10. Estevez J., Garcya P., Dyaz J. "Anomaly detection methods in wired networks: a survey and taxonomy." Computer Networks, том.27 – №.16, 2004, с. 1569–84.

11. Garcıa T., Dıaz V., Macia F., Vazquezb. "Anomaly-based network intrusion detection." Computers and security, том 28, 2009, с. 18–28.

12. Omar S., Ngadi A., Jebur H. "Machine Learning Techniques for Anomaly Detection: An Overview." International Journal of Computer Applications, том 79, № 2, 2013, с. 33–41.

13. GitHub - yzhao062/SUOD. "An Acceleration System for Large-scale Unsupervised Heterogeneous Outlier Detection (Anomaly Detection)." [Online]. Available: https://github.com/yzhao062/SUOD

14. Palo Alto Networks. "UTM." [Online]. Available: https://www.paloaltonetworks.com/cyberpedia/what-is-an-intrusion-preventionsystem-ips

15. ELKI Data Mining Framework. [Online]. Available: https://elki-project.github.io

16. GitHub-Markus-Go/rapidmineranomalydetection. "RapidMiner Extension for Anomaly Detection." [Online]. Available: https://github.com/Markus-Go/rapidmineranomalydetection

17. TechTarget. "IPS (Intrusion Prevention System)." [Online]. Available: https://www.techtarget.com/searchsecurity/definition/intrusion-prevention

18. cran.r-project. [Online]. Available: https://cran.r-project.org/view=AnomalyDetection

19. r-project. [Online]. Available: http://www.r-project.org/

20. dsmi-lab-ntust.github.io. "AnomalyDetectionToolbox." [Online]. Available: https://dsmi-lab-ntust.github.io

21. GitHub-datamllab/tods. "TODS: An Automated Time-series Outlier Detection System." [Online]. Available: https://github.com/datamllab/tods

22. GitHub - earthgecko/skyline. "Anomaly detection." [Online]. Available: https://github.com/earthgecko/skyline

23. GitHub - tsurubee/banpei. "Banpei. Anomaly detection library based on singular spectrum transformation (sst)." [Online]. Available: https://github.com/tsurubee/banpei

24. arxiv.org/abs/1802.04431. "Telemanom." [Online]. Available: https://arxiv.org/abs/1802.04431

25. GitHub - KDD-OpenSource/DeepADoTS. "Repository of the paper 'A Systematic Evaluation of Deep Anomaly Detection Methods for Time Series'." [Online]. Available: https://github.com/KDD-OpenSource/DeepADoTS

26. GitHub - numenta/NAB. "The Numenta Anomaly Benchmark." [Online]. Available: https://github.com/numenta/NAB

27. GitHub - twitter/AnomalyDetection. "Anomaly Detection with R." [Online]. Available: https://github.com/twitter/AnomalyDetection

28. r-project. "CRAN - Package anomalize." [Online]. Available: https://r-project.org

29. Open Distro. "Real Time Anomaly Detection in Open Distro for Elasticsearch." [Online]. Available: https://opendistro.io

30. GitHub - MentatInnovations/datastream.io. "An open-source framework for real-time anomaly detection using Python, Elasticsearch and Kibana." [Online]. Available: https://github.com/MentatInnovations/datastream.io

31. stonybrook.edu. "ODDS – Outlier Detection DataSets." [Online]. Available: https://stonybrook.edu

32. dataverse.harvard.edu. "Розмір даних для виявлення аномалій без нагляду." [Online]. Available: https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/OPQMVF

33. ir.library.oregonstate.edu. "Контрольні показники мета-аналізу виявлення аномалій." [Online]. Available: https://ir.library.oregonstate.edu/concern/datasets/47429f155

34. GitHub - waico/skab. "Skoltech Anomaly Benchmark." [Online]. Available: https://github.com/waico/skab

35. Network Attacks and Network Security Threats [Електронний ресурс]. – Режим доступу до ресурсу: https://www.cynet.com/network-attacks/networkattacks-

and-network-security-threats/ 36. A Python Toolbox for Scalable Outlier Detection (Anomaly Detection) [Електроний ресурс]. – Режим доступа до ресурса: https://GitHubyzhao062/pyod

37. Streaming Anomaly Detection Framework in Python (Outlier Detection for Streaming Data) [Електроний ресурс]. – Режим доступа до ресурса: https://GitHub - selimfirat/pysad:

38. Novelty and Outlier Detection – scikit-learn 1.0.1 documentation [Електроний ресурс]. – Режим доступа до ресурса: https://scikit-learn.org

39. Никишова А. В. Интеллектуальная система обнаружения атак на основе многоагентного подхода // Вестник Волгоградского государственного университета. Серия 10. Инновационная деятельность.. – 2011. – № 5. – С. 35–37

40. A Comparative Study of Rivest Cipher Algorithms / International Journal of Information & Computation Technology. ISSN 0974–2239 Volume 4, Number 17 (2014), pp. 1831–1838

41. Бойченко С. В., Іваненко О. В. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету. – Київ: НАУ, 2017. – 63 с.

42. Единая система конструкторской документации. Текстовые документы : ГОСТ 2.106-96. – [Дата введения 1997–07–01]. – М.: Стандартинформ, 1997. – 39 с. – (Межгосударственный стандарт).

Додаток А

```
#!/usr/bin/env python
# coding: utf-8

# # Importing the required packages

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.utils import resample
import seaborn as sns
from urllib.parse import urlparse
import tldextract
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix,classification_report
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import AdaBoostClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.model_selection import learning_curve


# # Loading the Train Dataset
```

```
df=pd.read_csv("url train dataset.csv",delimiter = ',',encoding = 'unicode_escape', low_memory = False)

df.head()

:

df.info()

df.describe()

# # Checking dataset distribution

df['label'].value_counts()

# # Plot of unbalanced distribution

df_ublabel0 = df[df.label==0]
df_ublabel1 = df[df.label==1]
df_ubclass = pd.concat([df_ublabel0,df_ublabel1])
```

```
plt.figure(figsize=(5, 8))
sns.countplot('label', data=df_ubclass)
plt.title('unbalanced Classes')
plt.show()


# # Resampling and balancing dataset



df_blabel1_resampled = resample(df_ublabel1, replace= False, n_samples=6251)
df_bsampled = pd.concat([df_blabel1_resampled,df_ublabel0])
df_bsampled.label.value_counts()


plt.figure(figsize=(5,8))
sns.countplot('label', data=df_bsampled)
plt.title('Balanced Classes')
plt.show()


# # Creating Features from train data


# Method to count number of dots
```

```
def countdots(url):
    return url.count('.')


# Method to count number of delimeters
def countdelim(url):
    count = 0
    delim=[';','_','?','=','&']
    for each in url:
        if each in delim:
            count = count + 1

    return count


# Is IP addr present as the hostname, let's validate

import ipaddress as ip #works only in python 3

def isip(url):
    try:
        if ip.ip_address(url):
            return 1
    except:
        return 0


#method to check the presence of hyphens
```

```
def isPresentHyphen(url):
    return url.count('-')



#method to check the presence of @

def isPresentAt(url):
    return url.count('@')



def isPresentDSlash(url):
    return url.count('//')



def countSubDir(url):
    return url.count('/')



def get_ext(url):

    root, ext = splitext(url)
    return ext
```

```
def countSubDomain(subdomain):
    if not subdomain:
        return 0
    else:
        return len(subdomain.split('.'))

]:


def countQueries(query):
    if not query:
        return 0
    else:
        return len(query.split('&'))



featureSet = pd.DataFrame(columns=('url','no of dots','presence of hyphen','len of url','presence of at','presence of double slash','no of subdir','no of subdomain','len of domain','no of queries','is IP','label'))


#get_ipython().system('pip install tldextract')



def getFeatures(url, label):
    result = []
```

```
url = str(url)

#add the url to feature set
result.append(url)

#parse the URL and extract the domain information
path = urlparse(url)
ext = tldextract.extract(url)

#counting number of dots in subdomain
result.append(countdots(ext.subdomain))

#checking hyphen in domain
result.append(isPresentHyphen(path.netloc))

#length of URL
result.append(len(url))

#checking @ in the url
result.append(isPresentAt(path.netloc))

#checking presence of double slash
result.append(isPresentDSlash(path.path))
#Count number of subdir
result.append(countSubDir(path.path))

#number of sub domain
result.append(countSubDomain(ext.subdomain))
```

```
    #length of domain name
    result.append(len(path.netloc))

    #count number of queries
    result.append(len(path.query))

    #Adding domain information

    #if IP address is being used as a URL
    result.append(isip(ext.domain))
    #result.append(get_ext(path.path))
    result.append(str(label))
    return result


for i in range(len(df)):
    features = getFeatures(df["domain"].loc[i],df["label"].loc[i])
    featureSet.loc[i] = features



featureSet.head()



featureSet.info()


# # Visualizing of Features
```

```
#


sns.set(style="darkgrid")
sns.distplot(featureSet[featureSet['label']=='0']['len                of
url'],color='green',label='Benign')
sns.distplot(featureSet[featureSet['label']=='1']['len                of
url'],color='red',label='Phishing')
plt.title('Distribution of URL Length')
plt.legend(loc='upper right')
plt.xlabel('Length of URL')
plt.show()

:


sns.set(style="darkgrid")
sns.distplot(featureSet[featureSet['label']=='0']['no                 of
dots'],color='green',label='Benign')
sns.distplot(featureSet[featureSet['label']=='1']['no                 of
dots'],color='red',label='Phishing')
plt.title('Distribution of Dots')
plt.legend(loc='upper right')
plt.xlabel('No of Dots')
plt.show()



sns.set(style="darkgrid")
```

```
        sns.distplot(featureSet[featureSet['label']=='0']['no of
subdir'],color='green',label='Benign')
        sns.distplot(featureSet[featureSet['label']=='1']['no of
subdir'],color='red',label='Phishing')
        plt.title('Distribution of Subdirectory')
        plt.legend(loc='upper right')
        plt.xlabel('No of Subdirectory')
        plt.show()


        sns.set(style="darkgrid")
        sns.distplot(featureSet[featureSet['label']=='0']['no of
subdomain'],color='green',label='Benign')
        sns.distplot(featureSet[featureSet['label']=='1']['no of
subdomain'],color='red',label='Phishing')
        plt.title('Distribution of Subdomain')
        plt.legend(loc='upper right')
        plt.xlabel('No of Subdomains')
        plt.show()



        sns.set(style="darkgrid")
        sns.distplot(featureSet[featureSet['label']=='0']['no of
queries'],color='green',label='Benign')
        sns.distplot(featureSet[featureSet['label']=='1']['no of
queries'],color='red',label='Phishing')
        plt.title('Distribution of Queries')
        plt.legend(loc='upper right')
```

```
        plt.xlabel('No of Queries')
        plt.show()


        # # Splitting the dataset to test and train

        #
        X=featureSet.iloc[:,1:11].values
        Y=featureSet.iloc[:,11].values
        X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state
=0)


        print(X_train.shape)
        print(Y_train.shape)

        print(X_test.shape)
        print(Y_test.shape)


        # # Training classifiers and testing without hyper parameter tuning

        print('Decision Tree Classifier')
        clf1 = tree.DecisionTreeClassifier()
        clf1.fit(X_train,Y_train)
        Y_pred = clf1.predict(X_test)
        print('Accuraccy : %f' % metrics.accuracy_score(Y_test, Y_pred))
```

```
print('Adaboost Classifier')
clf2 = AdaBoostClassifier()
clf2.fit(X_train,Y_train)
Y_pred = clf2.predict(X_test)
print('Accuraccy : %f' % metrics.accuracy_score(Y_test, Y_pred))


print('Logistic Regression Classifier')
clf3 = LogisticRegression()
clf3.fit(X_train,Y_train)
Y_pred = clf3.predict(X_test)
print('Accuraccy : %f' % metrics.accuracy_score(Y_test, Y_pred))

print('Gaussian NB Classifier')
clf4 = GaussianNB()
clf4.fit(X_train,Y_train)
Y_pred = clf4.predict(X_test)
print('Accuraccy : %f' % metrics.accuracy_score(Y_test, Y_pred))

print('KNN Classifier')
clf5 = KNeighborsClassifier()
clf5.fit(X_train,Y_train)
Y_pred = clf5.predict(X_test)
print('Accuraccy : %f' % metrics.accuracy_score(Y_test, Y_pred))


# # Hyper parameter tuning using GridSearch CV
```

```
print('Decision Tree Classifier')
param_grid = {"criterion" : ["gini", "entropy"],
              "max_depth": [3,5,20,30],
              "splitter" : ["best","random"]
             }
griddt = GridSearchCV(estimator=clf1, param_grid=param_grid)
griddt.fit(X_train,Y_train)
print(griddt)
# summarize the results of the grid search
print(griddt.best_score_)
print(griddt.best_estimator_.max_depth)
print(griddt.best_estimator_.criterion)
print(griddt.best_estimator_.splitter)


print('Adaboost Classifier')
param_grid = {"learning_rate" : [1,2,3,5,6],
              "n_estimators": [5,10,15,25,50],
              }
griddt = GridSearchCV(estimator=clf2, param_grid=param_grid)
griddt.fit(X_train,Y_train)
print(griddt)
# summarize the results of the grid search
print(griddt.best_score_)
print(griddt.best_estimator_.learning_rate)
print(griddt.best_estimator_.n_estimators)


print('Logistic Regression Classifier')
```

```
param_grid = {"penalty" : ["l1","l2"],
            "C": [0.1,0.5,1,1.5],
            }
griddt = GridSearchCV(estimator=clf3, param_grid=param_grid)
griddt.fit(X_train,Y_train)
print(griddt)
# summarize the results of the grid search
print(griddt.best_score_)
print(griddt.best_estimator_.penalty)
print(griddt.best_estimator_.C)


print('KNN Classifier')
param_grid = {"n_neighbors" : [1,2,3,4,5],
            "weights": ["uniform","distance"],
            }
griddt = GridSearchCV(cv=5,estimator=clf5, param_grid=param_grid)
griddt.fit(X_train,Y_train)
print(griddt)
# summarize the results of the grid search
print(griddt.best_score_)
print(griddt.best_estimator_.n_neighbors)
print(griddt.best_estimator_.weights)


# # Training and validating the classifiers with tuned hyper parameters using learning curves

print('Decision Tree Classifier')
```

```
clf1 = tree.DecisionTreeClassifier(max_depth =30 , criterion = 'gini', splitter = 'random')
clf1.fit(X_train,Y_train)
scores_1 = cross_val_score(clf1, X_train, Y_train, cv=5)
print("Accuracy: %f (+/- %0.2f)" % (scores_1.mean(), scores_1.std() * 2))
train_sizes, train_scores,validation_scores = learning_curve(clf1,X_train,Y_train,cv=5,scoring='accuracy', n_jobs=-1,train_sizes=np.linspace(0.01, 1.0, 50)
)
train_mean = np.mean(1-train_scores, axis=1)
validation_mean = np.mean(1-validation_scores, axis=1)
plt.plot(train_sizes, validation_mean, label = 'Validation error')
plt.plot(train_sizes, train_mean, label = 'training error')
plt.legend()



print('Adaboost Classifier')
clf2 = AdaBoostClassifier(learning_rate = 1, n_estimators = 50)
clf2.fit(X_train,Y_train)
scores_2 = cross_val_score(clf2, X_train, Y_train, cv=5)
print("Accuracy: %f (+/- %0.2f)" % (scores_2.mean(), scores_2.std() * 2))
train_sizes, train_scores,validation_scores = learning_curve(clf2,X_train,Y_train,cv=5,scoring='accuracy', n_jobs=-1,train_sizes=np.linspace(0.01, 1.0, 50)
)
train_mean = np.mean(1-train_scores, axis=1)
validation_mean = np.mean(1-validation_scores, axis=1)
plt.plot(train_sizes, train_mean, label = 'training error')
```

```
        plt.plot(train_sizes, validation_mean, label = 'Validation error')
        plt.legend()


        print('Gaussian NB Classifier')
        clf4 = GaussianNB()
        clf4.fit(X_train,Y_train)
        scores_4 = cross_val_score(clf4, X_train, Y_train, cv=5)
        print("Accuracy: %f (+/- %0.2f)" % (scores_4.mean(), scores_4.std() * 2))
        train_sizes, train_scores,validation_scores = learning_curve(clf4,X_train,Y_train,cv=5,scoring='accuracy', n_jobs=-1,train_sizes=np.linspace(0.01, 1.0, 50)
        )
        train_mean = np.mean(1-train_scores, axis=1)
        validation_mean = np.mean(1-validation_scores, axis=1)
        plt.plot(train_sizes, train_mean, label = 'training error')
        plt.plot(train_sizes, validation_mean, label = 'Validation error')
        plt.legend()

        print('KNN Classifier')
        clf5 = KNeighborsClassifier(n_neighbors = 4, weights = 'distance')
        clf5.fit(X_train,Y_train)
        scores_5 = cross_val_score(clf5, X_train, Y_train, cv=5)
        print("Accuracy: %f (+/- %0.2f)" % (scores_5.mean(), scores_5.std() * 2))
        train_sizes, train_scores,validation_scores = learning_curve(clf5,X_train,Y_train,cv=5,scoring='accuracy', n_jobs=-1,train_sizes=np.linspace(0.01, 1.0, 50)
        )
        train_mean = np.mean(1-train_scores, axis=1)
```

```
validation_mean = np.mean(1-validation_scores, axis=1)
plt.plot(train_sizes, train_mean, label = 'training error')
plt.plot(train_sizes, validation_mean, label = 'Validation error')
plt.legend()


# # Boxplot of error vs classifiers

# # Reducing the variance of best classifier (DT)


print('Decision Tree Classifier')
clf1 = tree.DecisionTreeClassifier(max_depth =7, criterion = 'gini', splitter = 'best')
clf1.fit(X_train,Y_train)
scores_1 = cross_val_score(clf1, X_train, Y_train, cv=10)
print("Accuracy: %f (+/- %0.2f)" % (scores_1.mean(), scores_1.std() * 2))
train_sizes, train_scores,validation_scores = learning_curve(clf1,X_train,Y_train,cv=5,scoring='accuracy', n_jobs=-1,train_sizes=np.linspace(0.01, 1.0, 50)
)
train_mean = np.mean(1-train_scores, axis=1)
validation_mean = np.mean(1-validation_scores, axis=1)
plt.plot(train_sizes, validation_mean, label = 'Validation error')
plt.plot(train_sizes, train_mean, label = 'training error')
plt.legend()


# # Testing the best classifier (DT)
```

```
print('Decision Tree Classifier')
Y_pred = clf1.predict(X_test)
from sklearn import metrics
print('Accuraccy : %f' % metrics.accuracy_score(Y_test, Y_pred))
cm = confusion_matrix(Y_test,Y_pred)
cr = classification_report(Y_test,Y_pred)
sns.heatmap(cm,annot=True,cbar=True,xticklabels='auto',yticklabels='auto')
print(cr)


# # Transfer learning of best classifier on new dataset using DT

dftl=pd.read_csv("url transfer dataset.csv",delimiter = ',',encoding = 'unicode_escape', low_memory = False)



dftl.describe()


featureSettl = pd.DataFrame(columns=('url','no of dots','presence of hyphen','len of url','presence of at','presence of double slash','no of subdir','no of subdomain','len of domain','no of queries','is IP','label'))

for i in range(len(dftl)):
    featurestl = getFeatures(dftl["domain"].loc[i], dftl["label"].loc[i])
    featureSettl.loc[i] = featurestl
```

```
featureSettl.info()


featureSettl.head()


X_testtl=featureSettl.iloc[:,1:11].values
Y_testtl=featureSettl.iloc[:,11].values
print(X_testtl.shape)
print(Y_testtl.shape)


print('Decision Tree Classifier')
Y_predtl = clf1.predict(X_testtl)
print('Accuraccy : %f' % metrics.accuracy_score(Y_testtl, Y_predtl))
cm = confusion_matrix(Y_testtl, Y_predtl)
cr = classification_report(Y_testtl, Y_predtl)
sns.heatmap(cm,annot=True,cbar=True,xticklabels='auto',yticklabels='auto')
print(cr)

# # Demo using DT

result = pd.DataFrame(columns=('url','no of dots','presence of hyphen','len of url','presence of at','presence of double slash','no of subdir','no of subdomain','len of domain','no of queries','is IP','label'))
results = getFeatures('https://www.google.com/', '')
result.loc[0] = results
```

```
result = result.drop(['url','label'],axis=1).values
print(clf1.predict(result))

result = pd.DataFrame(columns=('url','no of dots','presence of hyphen','len of url','presence of at','presence of double slash','no of subdir','no of subdomain','len of domain','no of queries','is IP','label'))
results = getFeatures('https://lms.vit.ac.in/login/index.php', '')
result.loc[0] = results
result = result.drop(['url','label'],axis=1).values
print(clf1.predict(result))

def check(url):
    result = pd.DataFrame(columns=('url','no of dots','presence of hyphen','len of url','presence of at','presence of double slash','no of subdir','no of subdomain','len of domain','no of queries','is IP','label'))
    results = getFeatures('url', '')
    result.loc[0] = results
    result = result.drop(['url','label'],axis=1).values
    return clf1.predict(result)[0]

check('https://lms.vit.ac.in/login/index.php')
```