Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2022р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти

на тему

“Дослідження та програмна реалізація системи виявлення вторгнень в промислову IoT-інфраструктуру”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-21М 1,4
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____Шовкопляс Ю.С.
«____» _____ 2022р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Якименко Н.М.
«____» _____ 2022р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 123 *"Комп'ютерна інженерія"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерна інженерія"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2022 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Шовкоплясу Юрію Станіславовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи виявлення вторгнень в промислову IoT-інфраструктуру*

2. Керівник роботи *Якименко Наталія Миколаївна, канд. фіз.-мат. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 19-13 від 17.08.2022 року

3. Строк подання студентом роботи до захисту *21.12.2022 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є дослідження та програмна реалізація системи виявлення вторгнень в промислову IoT-інфраструктуру*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Економічна ефективність розробленої програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2022	14.11.2022
Охорона праці	Оришака О.В.	06.10.2022	16.11.2022

7. Дата видачі завдання « 6 » вересня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2022 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2022 р.	
3.	Розробка моделі компонента	20.10.2022 р.	
4.	Розробка структур даних	25.10.2022 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2022 р.	
6.	Програмування алгоритмів	10.11.2022 р.	
7.	Розрахунок економічної ефективності	13.11.2022 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2022 р.	
9.	Оформлення ПЗ	17.11.2022 р.	
10.	Попередній захист роботи	10.12.2022 р.	

Дата видачі завдання
« 6 » вересня 2022 р.

Підпис керівника
_____ Якименко Н.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2022 р.

Підпис здобувача
_____ Шовкопляс Ю.С.
(прізвище та ініціали)

АНОТАЦІЯ

Шовкопляс Ю.С. Дослідження та програмна реалізація системи виявлення вторгнень в промислову IoT-інфраструктуру. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2022.

В даній магістерській роботі розроблено програмне забезпечення, яке призначено для системи виявлення вторгнень в промислову IoT-інфраструктуру з використанням алгоритму машинного навчання.

Метою розробки є дослідження та програмна реалізація системи виявлення вторгнень в промислову IoT-інфраструктуру з використанням алгоритму машинного навчання.

Об'єктом дослідження є методи системи виявлення вторгнень в промислову IoT-інфраструктуру з використанням алгоритму машинного навчання.

Предметом дослідження є методи виявлення вторгнень в промислову IoT-інфраструктуру.

Методи дослідження базуються на методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи виявлення вторгнень в промислову IoT-іфраструктуру.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows XP/Vista/7/8/10.

Програму розроблено в середовищі PyCharm Community Edition 2022.2.3.

Ключові слова: комп'ютерна інженерія, система виявлення вторгнень, машинне навчанння

ABSTRACT

Shovkoplias Y.S. Research and software implementation of the system for detecting intrusion into the industrial IoT infrastructure. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2022.

In this master's thesis, software has been developed that is designed for the system for detecting intrusion into the industrial IoT infrastructure using machine learning algorithm.

The purpose of the development is to study and software implementation of the system for detecting intrusion into the industrial IoT infrastructure using machine learning algorithm.

The object of the study is the process of creation system for detecting intrusion into the industrial IoT infrastructure using machine learning algorithm.

The subject of the study is the detection methods of system for detecting intrusion into the industrial IoT infrastructure using machine learning algorithm

Research methods are based on information protection methods, mathematical statistics methods, and software development methods.

The result of the work is a software implementation of the system for detecting intrusion into the industrial IoT infrastructure using machine learning algorithm.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

Developed user-friendly interface. Instructions for working with software are given.

The program can be used on an IBM PC PC with Windows XP / Vista / 7/8/10.

The program is developed in the PyCharm Community Edition 2022.2.3. environment.

Keywords: computer engineering, intrusion detection system, machine learning

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

СВВ	–	Система виявлення вторгнень
IoT	–	Інтернет речей
IT	–	Інформаційні технології
DDoS	–	Distributed Denial of Service
TCP	–	Протокол керування передаванням
IP	–	Міжмережевий протокол
ОС	–	Операційна система
HIDS	–	Система виявлення вторгнень на основі хоста
ANN	–	Штучні нейронні мережі
AIM	–	Штучні імунні системи
RL	–	Навчання з підкріпленням
ПЗ	–	Програмне забезпечення
DoS	–	Denial of Service
LR	–	Логістична регресія
NB	–	Байєсівський алгоритм
KNN	–	алгоритм knn-сусідів
DT	–	дерево вибору
AB	–	AdaBoost
RF	–	Випадковий ліс
SSH	–	Безпечна оболонка

ВСТУП

Актуальність теми. Актуальність дослідження полягає в тому, що все більше компаній почало використовувати пристрої Інтернету речей. Зростаюча популярність IoT надає більш широкі покращення для планування або автоматизації роботи на підприємстві. Інтернет речей дозволяє поєднати в одну мережу та керувати множиною пристроїв, що виконують забезпечення збору, аналізу та передачу даних. Це сприяло появленню більшої кількості рішень на ринку IoT технологій, що впливає на потенціальний ріст вразливостей цих мереж. Найпоширенішим методом атаки на пристрої IoT є ботнети. Наприклад, такі ботнети як Mirai, користуються незахищеністю пристроїв IoT для здійснення DDoS атак на критичну інфраструктуру.[8]

Це привело до того, що зросла кількість розробки нових методів для автоматичного реагування на атаки IoT. Ці системи відстежують мережеву активність підключених пристроїв IoT і тримають фокус на попередженні про шкідливу або підозрілу активність. Системи виявлення вторгнень також може виявляти та протидіяти аномальним діям, перекриваючи зловмиснику доступ до IoT пристроїв.

Найпоширенішими DDoS атаками на сьогоднішній день є DNS, UDP, SYN та TCP атаки. Через обмеження в обробці, пам’яті та неоднорідному характері пристроїв Інтернету речей реалізація систем виявлення вторгнень в таких пристроях є складною та актуальною проблемою.

Таким чином, система виявлення вторгнень виявляє мережеві атаки на вразливі програми та служби, несанкційований вхід в систему та отримання доступу до конфіденційних документів , атаки на основі хостів, а також зараження зловмисним програмним забезпеченням.

Мета й завдання дослідження. Метою роботи є розроблення програмного забезпечення системи виявлення вторгнень в промислову IoT-інфраструктуру. Для

досягнення поставленої мети визначена програма дослідження, що складається з наступних пунктів:

- Огляд існуючих систем виявлення вторгнень в промислову IoT-інфраструктуру.

- Дослідження виявлення вторгнень в промислову IoT-інфраструктуру.

- Програмна реалізація системи виявлення вторгнень в промислову IoT-інфраструктуру.

Об'єктом дослідження є процес створення системи виявлення вторгнень в промислову IoT-інфраструктуру.

Предметом дослідження є методи виявлення вторгнень в промислову IoT-інфраструктуру.

Методи дослідження базуються на методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступний результат:

– Розроблено вітчизняний продукт системи виявлення вторгнень в промислову IoT-інфраструктуру, який має більш широкі можливості, на відміну від існуючих аналогів. Ці можливості полягають у застосуванні декількох алгоритмів навчання моделі даних та їх порівняння один із одним, що зменшує вірогідність помилкової класифікації подій.

– Удосконалено метод забезпечення безпеки системи промислової IoT-інфраструктури завдяки застосуванню машинного навчання для виявлення вторгнень. Реалізація зазначеного метода дозволяє зменшити завантаженість ресурсів комп'ютерної системи за рахунок спрямування безпечних неоднорідних інформаційних потоків на безпечні та менш завантажені ресурси.

Практична цінність отриманих результатів полягає в тому, що розроблене програмне забезпечення дозволить успішно вирішувати задачі виявлення вторгнень в промислову IoT-інфраструктуру.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Таким чином, виходячи з вище перерахованого, програмне забезпечення системи виявлення вторгнень в IoT-інфраструктуру, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній магістерській роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 **Призначення системи**

Призначенням системи є реалізація виявлення вторгнень в IoT-інфраструктуру, за рахунок програмної реалізації алгоритму машинного навчання. Цей алгоритм буде забезпечувати захист інформації від вторгнень.

Методи машинного навчання мають за ціль взаємодію з пристроями, що використовують комунікаційні технології, наприклад BLE, LoRaWAN, 6LoWPAN і так далі. В більшості випадків системи виявлення вторгнень роблять прогнози, посилаючись на повідомлення, що отримуються від пристроїв Інтернету речей, користуючись керувальною інформацією конкретних технологій, наприклад як перевірка відповідності протоколу. Системи виявлення вторгнень, які націлені на Інтернет речей бувають IoT-специфічними та IoT-агностичними.[2]

На інформаційні мережі і системи проводяться на постійній основі кібер-атаки. Антивірусів та брандмауерів недостатньо для виявлення цих атак, тому що вони лише захищають поверхнево системи і мережі комп’ютерів.

В мережі Інтернет є дуже велика кількість шкідливого софту, що є в свою чергу безкоштовним, а саме: слеммери та слеппери і т.д. Конкуренти можуть користуватися послугами хакерів для того, щоб нашкодити іншій компанії.

Системи виявлення вторгнень виконують наступні завдання: аналіз джерел інформації та відповідна реакція, відносно результату аналізу. Щоб виконати ці завдання система виявлення вторгнень має здійснити:

- моніторинг та аналіз активності користувачів;
- проводити аудит конфігурації системи та її слабких місць;
- перевіряти цілісність найважливіших файлів системи та файли даних;
- проводити статистичний аналіз стану системи, що порівнюється із станами, які були при відомих атаках;

- здійснювати аудит операційної системи (ОС).[3]

Системи виявлення вторгнень мають три основні компоненти:

- мережева система виявлення вторгнень (NIDS): вона виконує аналіз трафіку через всю підмережу і дозволяє відповідати трафіку, через вже відомі атаки у бібліотеці;

- система виявлення вторгнень у мережу (NNIDS): вона схожа на мережеву, але проводить контроль трафіку на одному хості;

- система виявлення вторгнень на основі хоста (HIDS): вона може виконувати "знімок" усього набору файлів в системі та робити порівняння з зображенням, що було зроблено перед цим. При наявності суттєвих відмінностей, система попереджує адміністратора.[4]

Тому, системи виявлення вторгнень є дуже необхідними для захисту інфраструктури Інтернету речей і все більше їх впроваджують в експлуатацію. Діяльність бізнесу в мережі Інтернет зросла, що приводить до запуску комп'ютерних мереж. Це робить компанії більш вразливими для зловмисників. Тому, питання захисту інформаційних систем та мереж стало дуже важливим. Системи виявлення вторгнень стали важливою частиною мережі організації, що відіграє роль для виявлення несанкційованого доступу до цих систем.[1]

1.2 Область застосування

Інтернет речей є дуже поширеним у транспортних, комунальних та виробничих організаціях, де мають місце використання датчиків. Також, Інтернет речей почав використовуватися для організації роботи пристроїв в сільському господарстві, домашньої автоматизації та промисловій інфраструктурі, що дало поштовх до впровадження деякими організаціями цифрових технологій.

IoT є корисним для фермерів тим, що може значно полегшити їм роботу. Наприклад, можна використовувати датчики для збору даних, а саме: кількість

опадів, температура, вологість і вміст ґрунту та багато інших факторів, що можуть допомогти автоматизувати сільськогосподарську техніку.[7]

Однією із важливих можливостей Інтернету речей є здатність спостерігати за операціями навколо інфраструктури. Наприклад, датчики можуть бути використані в спостереженні за подіями або змінами в структурних будівлях, мостах та інших частинах інфраструктури. Це дозволяє значно зекономити кошти та час, які витрачались на ці операції, також дозволяє наявність змін в робочому процесі якості життя та відмовитись від використання паперів в робочому процесі.

Пристрої домашньої автоматизації можуть використовувати технології Інтернету речей в таких процесах як керування та моніторинг за електричними і механічними системами в будівлі. В масштабах міста це може допомогти громадянам зменшити кількість споживання енергії та кількість відходів.

Інтернет речей має відношення до будь-якої галузі, що включає в себе фінанси, виробництво, охорону здоров'я та роздрібну торгівлю.[5]

IoT поєднує мільярди пристроїв з мережею Інтернет та може передбачати використання великої кількості точок даних, забезпечуючи для них захист. Безпека та конфіденційність Інтернету речей називаються основними проблемами через широку поверхню атаки.

Одна із найвідоміших атак на IoT була атака 2016-го року, причиною цьому став Mirai – ботнет, що проник до постачальника серверів доменних імен Dyn та зміг заблокувати велику кількість веб-сайтів на значний період часу в такому типу атак як DDoS. Через слабкий захист пристроїв Інтернету речей, зловмисники змогли отримати доступ до мережі.

Через те, що пристрої IoT дуже взаємопов'язані між собою, то для хакера достатньо лише знайти одну вразливість для отримання та можливості маніпулювати усіма даними, що робить їх непридатними для використання. Не оновлюючи свої пристрої на регулярній основі, виробники роблять їх більш вразливими для хакерів.[50]

Окрім того, більшість підключених пристроїв може запрошувати від користувача ввести свої особисті дані, а саме, ім’я, номер телефону, адресу, вік та іншу важливу інформацію про користувача, яка може в будь-який момент попасти до рук кіберзлочинців, що будуть використовувати її для шантажу або продажу цих даних.

Але зловмисники не є єдиною загрозою для Інтернету речей. Конфіденційність також є проблемою для тих, хто користується IoT. Це можуть бути компанії, які розповсюджують пристрої Інтернету речей, вони можуть продати інформацію про користувачам іншим компаніям. Також це впливає на створення ризику для критично важливої інфраструктури, а саме транспорт, фінансові послуги та електроенергію.[6]

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Розглянемо програмне забезпечення, що виконує функцію виявлення вторгнень в IoT-інфраструктурі.

До програмного забезпечення, для виявлення вторгнень в промислову IoT-інфраструктуру відносять:

- CrowdStrike Falcon.
- Snort.
- OSSEC.
- Suricata.
- Zeek.
- Security Onion.
- AIDE.
- Samhain.

CrowdStrike Falcon

Falcon – це платформа CrowdStrike, ціллю якого є запобігання зламів використовуючи уніфікований набір хмарних технологій, які можуть запобігати усі типи атак, включаючи зловмисне програмне забезпечення і т.д. Хоча, сьогодні хакери та зловмисники можуть використовувати не тільки зловмисне програмне забезпечення для проникнення в системи організацій, віддаючи перевагу більше таким видам злому як: нульові методи та дні, експлойти, що є дуже складним для виявлення, також можуть викрасти облікові дані та використовувати інструменти, що вже є частиною операційної системи чи середовища жертви, наприклад PowerShell. CrowdStrike Falcon реагує на спроби зламу потужним, але легким

рішенням, що об’єднує в собі антивірус (NGAV), виявлення кінцевих точок і реагування на них (EDR), проведення розвідки для виявлення кіберзагроз, наявністю керованого пошуку потенційних загроз та гігієни безпеки. Всі ці фактори знаходяться в одному маленькому та легкому датчику, що керується хмарою.[10]

Переваги і недоліки CrowdStrike Falcon:

Переваги:

- можливість керувати тисячами кінцевих точок;
- є free trial версія для ознайомлення;
- інтегрована розвідка загроз.

Недоліки:

- не є ефективним проти деяких загроз;
- має ціну вище, ніж у аналогів;
- має обмежені можливості звіту для основного продукту.[11]

Snort

Snort – це мережева система виявлення вторгнень з відкритим кодом, розроблена Мартіном Решем. На даний момент розробкою і підтримкою Snort займаються Cisco.

Snort є сніфером пакетів, що здатний відстежувати мережевий трафік, проводячи детальне вивчення кожного пакету, для виявлення підозрілих аномалій або небезпечного корисного навантаження. На протязі довгого відрізку часу, є лідером серед корпоративних інструментів для виявлення і запобігання вторгнень, також є доступним для операційних систем Unix, Linux та Windows.

Основна функція Snort захоплювати пакети бібліотеки(libpcap). Libpcap — це інструмент, що має широке використання в аналізаторах трафіку адреси протоколу керування передачею/протоколу Інтернету, пошуку та аналізу вмісту для реєстрації пакетів, аналізу протоколів і їх порівняння змісту та аналіз трафіку в режимі реального часу.

Також, користувачі можуть налаштувати Snort як реєстратор пакетів, сніффер, наприклад Wireshark або TCPdump, або ж як метод для запобігання вторгнення в мережу.

Snort відстежує мережевий трафік та виконує порівняння його із набором правил, що задає користувач. Snort застосовує ці правила до трафіку, що відстежується та сповіщає користувача про аномалії в активності мережі.

Завдяки цьому, він може ідентифікувати різні методи атак на системи кібербезпеки, а саме: відмова в обслуговуванні, відбитки ОС, переповнення буфера, атаки на звичайний інтерфейс шлюзу, невидиме сканування портів і зондування блоку повідомлень серверу.

Коли Snort знаходить аномалії в поведінці, він надсилає сповіщення в режимі реального часу до Syslog, як брандмауер та поміщає їх в окремий файл сповіщень чи використовує спливаюче вікно.[12]

Плюси та мінуси використання Snort:

Переваги:

- Snort виконує швидку інсталяцію та працює в мережі;

- написання правил Snort не є складним;

- Snort є безкоштовною і досить ефективною системою виявлення вторгнень.

Недоліки:

- при наявності у Snort гнучкості, він не має деяких функцій, що наявні у комерційних аналогах;

- адміністратор має створити індивідуальні методи звітності та журналювання;

- правила Snort мають бути створені ретельно. Це потрібно для зменшення кількості помилкових спрацьовувань генерованої інформації та зменшення кількості зареєстрованої інформації.

Використовувати Snort можна різними способами, а саме: як реєстратор пакетів, сніфер пакетів та систему виявлення вторгнень. Через наявність

можливості використовувати набори правил для моніторингу IP-пакетів, Snort є практичним рішенням для мереж малого та середнього розміру.[13]

OSSEC

OSSEC — це широко використовувана служба виявлення вторгнень на основі хоста, що сповіщає про багато факторів, наприклад про модифікацію системи. Він має дуже потужний механізм аналізу та кореляції, перевірку цілісності файлів, централізоване застосування політики, моніторинг реєстру Windows, виявлення руткітів, інтегруючи аналіз журналів та попереджує в режимі реального часу і активну відповідь. OSSEC може працювати на великій кількості операційних систем, а саме: Windows, Linux, MacOS, FreeBSD, OpenBSD та Solaris.[14]

Переваги OSSEC:

- надає сповіщення в режимі реального часу відносно аномалій та дозволяє швидко реагувати;

- аналіз журналів приймає їх у наступних форматах: бази даних, веб-сервери та FTP-сервери;

- здійснює ефективний збір системної інформації та діє як система інвентаризації;

Недоліки OSSEC:

- при оновленні версії OSSEC може статися неузгодження між правилами;

- наявність можливості помилки координації з попередніми спільними ключами;

- відсутність інформаційної панелі моніторингу негативно впливає на візуалізацію загроз.[15]

Security Onion

Security Onion — це безкоштовний дистрибутив Linux із відкритим кодом для виявлення вторгнень, моніторингу безпеки та керування журналами підприємства. Він включає в себе велику кількість інструментів, а саме: Squert, CyberChef, NetworkMiner, Kibana, Snort, Wazuh, Suricata, Bro, Sguil, Logstash і т.д. Є простим у

експлуатації майстер налаштувань, що надає можливість створити велику кількість датчиків для підприємства лише за декілька хвилин.

Logstash виконує збір всіх журналів, Elasticsearch зберігає їх у файли, для того, щоб користувач мав до них доступ, а Kibana дозволяє аналізувати їх і робить візуалізацію того, що відбувається на основі Security Operation Center. Kibana включає в себе можливість повністю перехопити всі пакети та пірнути в будь які точки інтересу передбачуваного інциденту безпеки.

Це дуже велика кількість інформації, яку необхідно прокопати для IoC, тому Security Onion супроводжується Sguil, що дає можливість для Security Operation Center бачити всі сповіщення від Snort, Wazuh і Suricata в одному місці і дозволяє перейти від сповіщення до певного перехоплення пакетів.

Тобто, Security Onion є безкоштовною системою виявлення вторгнень, які неважко знайти. Це чудовий повчальний засіб як для студента, так і для персоналу.[16]

Переваги і недоліки Security Onion:

Переваги:

- простий інтерфейс користувача;
- є безкоштовним програмним забезпеченням;
- має активну спільноту, що регулярно оновлює його.

Недоліки:

- для великих мереж потрібно більше ресурсів;
- адміністратор має навчитися добре використовувати інструменти, щоб отримати повну віддачу від програмного забезпечення;
- має високі накладні витрати на знання.[17]

AIDE

AIDE створює бази даних, використовуючи правила регулярних виразів, які знаходяться у файлі конфігурації. Після того, як була проведена ініціалізація бази даних, її можна використовувати для перевірки цілісності файлів. Це програмне забезпечення має декілька алгоритмів дайджесту повідомлень, що

використовуються для перевірки цілісності файлів. Всі звичайні атрибути файлів можуть бути перевірені на невідповідності.

AIDE може контролювати файли, які були змінено або модифіковано нещодавно. Ці файли або каталоги можна відстежувати, при спробах їх змінити.

AIDE захищено SELinux. SELinux забезпечує захист процесів AIDE за допомогою обов'язкового доступу до контролю. Він може визначати який тип процесу в кожному із процесів, що виконується в системі. Політика SELinux AIDE дуже гнучка, що надає можливість користувачам виконувати налаштування процесів AIDE максимально безпечними методами.[18]

Переваги і недоліки AIDE:

Переваги:

- підтримка регулярних виразів для вибіркового включення або включення файлів і каталогів для моніторингу;

- автономний статичний двійковий файл для зручності конфігурацій моніторингу клієнт/сервер.

Недоліки:

- має обмежену функціональність;

- призначений для одного хоста, не є кластером;

- незашифрована база даних.[19]

Samhain

Система виявлення вторгнень Samhain забезпечує виконання перевірки цілісності файлів, спостереження і аналіз файлів журналу, а також моніторинг портів, виявлення прихованих процесів, руткітів та фальшивих виконуваних файлів SUID.

Samhain був розроблений для спостереження за кількома хостами як з однаковими так і різними операційними системами, забезпечуючи обслуговування і централізоване журналювання. Ця система виявлення вторгнень також працює і для одного хоста.

Samhain – мультиплатформна програма з відкритим кодом для систем POSIX, а саме: Windows, Cygwin, Linux, Unix.

Особливості Samhain полягають у тому, що він має наступні функції:

Централізований моніторинг. Архітектура клієнт/сервер дозволяє централізовано вести журнал, зберігати бази даних і конфігурації даних та оновлювати бази даних.

Веб-консоль керування. Веб-консоль Beltane, може відстежувати активність як клієнта так і сервера, а також переглядати звіти клієнтів і оновлювати бази даних.

Стійкість тамперу. Samhain може запропонувати PGP базу даних та конфігураційні файли, прихований режим і декілька інших функцій, щоб захистити свою цілісність.[20]

Переваги і недоліки Samhain:

Переваги:

- є open source продуктом;
- має дуже просту єдину систему;
- дозволяє централізовано змінювати бази даних;
- клієнти можуть надсилати звіт на сервер.

Недоліки:

- є проблема початкової безпеки бази даних;
- накладні витрати на мережу режим клієнт-сервер;
- проблеми центральної конфігурації зміни після оновлень;
- включає численні параметри.[21]

Suricata

Suricata – це система виявлення вторгнень, що має відкритий код. Вона може діяти як система виявлення вторгнень так і система запобігання вторгненням. Ця програма була розроблена фондом відкритої інформаційної безпеки (OSIF) і є безкоштовною. Її використовують як малі так і великі підприємства. Ця система

використовує набір правил і мову підписів для виявлення загрози і її запобігання. Suricata може працювати на Windows, MacOS, Linux та Unix.

Suricata може як вживати заходів відносно події так і блокувати трафік та в свою чергу, добре справляється з глибокою перевіркою пакетів. Ці фактори роблять Suricata майже ідеальним для стандартного спостереження за безпекою компанії.

Ця програма є легкою, дешевою та надає розуміння того, що відбувається в мережі з точки зору безпеки.

Основною відмінністю Suricata від Snort є те, що Suricata – багатопотоковий інструмент. Тому він може використовувати одразу декілька ядер, що балансує навантаження на систему. Це дає можливість обробляти більше даних, не повертаючись до кількості запроваджених правил. Системи керування інформацією про безпеку та подіями (SIEM) також можуть використовувати вихідні дані Suricata для того, щоб вдосконалити правила і процеси виявлення вторгнень.[22]

Переваги і недоліки Suricata:

Переваги:

- є сумісним з вже існуючими мережевими компонентами;

- є багатопотоковим;

- можливість налаштувати та використовувати ті самі набори правил як і в Snort.[25]

Недоліки:

- створює більше помилкових тривог на момент виявлення;

- використання великої кількості системних і мережевих ресурсів.[23]

Zeek

Zeek – пасивний аналізатор мережевого трафіку з відкритим кодом. Багато операторів користуються Zeek як монітором безпеки мережі (NSM), щоб підтримати розслідування підозрілої активності. Він також підтримує дуже широкий спектр аналізу трафіку за межами сфери безпеки, включаючи вимірювання рівня продуктивності та усунення неполадок.

Перша і важлива перевага, що отримає новий користувач після початку користування Zeek, це великий набір журналів, які описують мережеву активність. Вони включають в себе не тільки вичерпний запис кожного з'єднання, а й стенограми прикладного рівня. До них відносять всі HTTP-сеанси з запитуваними URL, заголовками ключів, відповідями серверів та MIME типами. За замовчуванням Zeek записує всю цю інформацію в журнали, що були добре структуровані, які в свою чергу розділені вкладками або ж в JSON-файли, які придатні для подальшої обробки за допомогою зовнішнього програмного забезпечення. Також користувачі мають можливість вибрати те, що зовнішні бази даних або продукти SIEM зберігали, використовували, обробляли та представляли дані для запитів.

Також, Zeek має вбудований функціонал для аналізу та виявлення, що включає в себе видобування файлів із HTTP-сеансів, виявлення шкідливого програмного забезпечення за допомогою зовнішніх реєстрів, звітів про вразливості версії даного програмного забезпечення, що були помічені в Інтернет-мережі та ідентифікацію популярних веб-сайтів.

Маючи такий потужний функціонал, Zeek має можливість бути повністю налаштованою та розширеною платформою для аналізу трафіку. Zeek дає користувачам повну мову сценаріїв для конкретних доменів, щоб мати можливість виконати завдання аналізу. Також, усі стандартні аналізи Zeek виконуються лише за допомогою сценаріїв, в ядрі системи немає закодованих конкретних аналізів.

Цей програмний засіб працює на звичайному апаратному забезпечені, а отже, не є дорогим рішенням. В багатьох аспектах, Zeek перевищує можливості деяких інструментів моніторингу мережі, які в свою чергу мають обмеження невеликим набором закодованих завдань аналізу. Zeek не є класичною системою виявлення вторгнень. При дотримані стандартних функцій мови сценаріїв, Zeek має значно ширше коло різних підходів до пошуку шкідливих дій. Вони включають в себе: виявлення семантичного неправильного використання, виявлення аномалій і аналіз поведінки.

Дуже багато сайтів розгортає Zeek для захисту своєї інфраструктури, що включає в себе лабораторії, університети, спільноти відкритих наук, великі корпорації, державні установи і т.д.

Zeek має підтримку високої продуктивності, підтримуючи масштабоване балансування навантаження. Центральна керуюча система виконує координацію процесу, синхронізуючи стан серверів та забезпечення надання операторам центрального інтерфейсу керування для налаштування та доступу до зведених журналів. Інтегрована структура керування Zeek та ZeekControl, здатна підтримувати налаштування кластерів із коробки.

Функції кластера Zeek підтримують як односистемні так і багатосистемні налаштування. Це є однією із переваг масштабованості Zeek. Наприклад, адміністратори можуть виконувати масштабування Zeek в одній системі на протязі великого відрізку часу, а потім, при необхідності додавати інші системи.

Отже, Zeek є добре оптимізованим для інтерпретації мережевого трафіку та створення журналів на основі цього трафіку. Але він не є оптимізованим для зіставлення байтів, також для користувачів, що шукають підходи до виявлення сигнатури, мають спробувати іншу систему для виявлення вторгнень. Zeek знаходиться у середовищі, що представляє компактні, але досить високої якості журнали мережі, створюючи краще розуміння мережевого трафіку та його використання.[24]

Виконавши огляд та аналіз переваг і недоліків вище наведених аналогів, було зроблено наступні висновки. ПЗ повинно мати більшу точність виявлення вторгнень та меншу кількість помилкових спрацьовувань, тому потрібно навчати одразу декілька методів і порівнювати їх результати для вибору кращого методу, що підходить заданій базі даних.

2.2 Обґрунтування вибору засобів для побудови системи мови програмування

Програмне забезпечення написане мовою Python. Ця мова програмування обрана виходячи з наступних міркувань. Python – це об’єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Високорівневі структури даних в Python у поєднанні із динамічною типізацією та динамічним зв’язуванням роблять з нього дуже привабливу для швидкого розроблення додатків мову програмування, а також для його використання як мови сценаріїв або поєднуваної мови для поєднання існуючих компонентів. Python має достатньо простий для вивчення синтаксис, підвищує читабельність коду і цим знижує ціну на обслуговування програми. Python має підтримку пакетів і модулів, що спонукає до повторного використання коду та модульності програми. Інтерпретатор Python та обширна стандартна бібліотека доступні в двійковому або вихідному вигляді і є безкоштовною та має можливість вільно поширювати для всіх основних платформ.

Перевага Python полягає в тому, що він має високу продуктивність і може її забезпечити. Оскільки ця мова програмування не має етапу компіляції, цикл редагування-тестування-налагодження відбувається дуже швидко. Виконувати налагодження коду програми на Python є достатньо легким процесом: помилки чи неправильний вхід не можуть спричинити помилку сегментації. Коли інтерпретатор знаходить помилку, він виконує виклик винятку. Якщо програма не може вловити виняток, то інтерпретатор виконує друк трасування стеку. Налагоджувач на рівні вихідного коду дозволяє перевірити глобальні та локальні змінні, оцінити довільні вирази, встановити контрольні точки, виконувати код програми покроково і т.д. Написаний на Python налагоджувач свідчить про інтроспективну силу цієї мови програмування. Але з іншого боку, досить часто найшвидшим способом налагоджувати програму можна за допомогою додавання кількох операторів друку до джерела: досить швидкий цикл редагування-тестування-налагодження може зробити цей підхід ефективнішим. [26]

Python став одним із основних елементів в data science, що дає можливість аналітикам даних та іншим експертам використовувати цю мову програмування для створення різних візуалізацій даних, проведення складних статистичних обчислень, створення різноманітних алгоритмів машинного навчання, аналізу та обробки даних і виконувати багато завдань, що пов'язані із даними.

Python може створювати досить широкий спектр візуалізацій даних, а саме: секторні діаграми, гістограми, 3D-графіки, стовпчасті та лінійні діаграми. Також в Python є дуже багато бібліотек, що дають можливість програмістам писати програми для аналізу даних і машинного навчання значно швидше і ефективніше.

Python також може автоматизувати завдання, якщо воно виконується неодноразово для більш ефективної роботи. Створення автоматизованих процесів в Python називається створенням сценаріїв. Створення сценаріїв в програмуванні часто використовують для перевірки помилок у декількох файлів, виконання нескладних математичних розрахунків, перетворення файлів та видалення однакових даних.

Python під час проведення розробки програмного забезпечення може допомогти в різних завданнях, а саме: відстеження помилок, контроль збірки та тестування. Деякі інструменти цієї мови програмування, що використовуються для проведення тестування програмного забезпечення включають в себе Requestium і Green.[27]

Сам по собі Python може надавати модулі та пакети для вивчення. Він також підтримує модульність програми та повторне використання коду. Працюючи з Python, варто звернути увагу на наступні допоміжні інструменти:

- Python 3.0, який був створений в 2008 році. Від попередніх оновлень, у Python 3 відбулися зміни в стилі кодування та сумісності. В наслідок чого він не міг підтримувати попередні оновлення. Повторювання синтаксису коду та надмірність зробили вивчення Python більш доступним і в свою чергу дозволяє вирішувати одні й ті самі завдання, але вже багатьма різними способами.[52]

- Інтегроване середовище розробки та навчання(IDLE) – стандартне середовище розробки в Python. В свою чергу, цей інструмент може забезпечити доступ до інтерактивного режиму в Python використовуючи вікно оболонки. Для користувачів є можливість використовувати IDLE, щоб створювати або редагувати існуючі файли Python за допомогою редактора файлів.

- PythonLauncher – це інструмент, що надає можливість розробникам виконувати запуск сценаріїв Python прямо із робочого столу. Для цього потрібно просто вибрати PythonLauncher як програму за замовчуванням, для того щоб відкрити будь-який сценарій в .py форматі.[53]

- Anaconda – це провідний дистрибутив із відкритим кодом для Python та R, що має в собі більше ніж 300 вбудованих бібліотек, які спеціально розробили для проектів Machine Learning. Основною метою Anaconda є спрощення розгортання та керування пакетами.

Python є досить економічно ефективним рішенням, коли користувачі додають безкоштовну стандартну бібліотеку чи інтерпретатор Python. Він є надзвичайно універсальним. Розробники мають можливість швидко задіяти цикл редагування-тестування-налагодження без необхідності компілювати код. Також є дуже багато інших причин, через які Python є більш практичним в написанні програмного забезпечення.[28]

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи виявлення вторгнень в IoT-інфраструктуру.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Для ідентифікації атаки на систему виявлення аномалій є важливим кроком. За основу виявлення аномалій береться класифікація подій, які відрізняються від нормальної поведінки. Програми виявлення вторгнень передбачають, що будь-які підозрілі події є підмножинами аномальних активностей.

Машинне навчання відіграє дуже важливу роль при створенні нормальних профілів та виявлення аномалій в системах виявлення вторгнень. Аномалії, що були виявлені, мають доступ до промаркованих даних, які є частиною нормальної поведінки системи. Але при зміні послуг або мережевого середовища, шаблони нормальної поведінки можуть змінитися. Наявність відмінностей між тестовими та навчальними даними призводить до великої кількості помилок при виявленні вторгнень у систему. Виявлення вторгнень, використовуючи методи машинного навчання без вчителя може подолати ці помилки. Через ці причини, на практиці, частіше використовуються саме ці методи та методи гібридного навчання.[29]

Системи виявлення вторгнень зазвичай призначені для моніторингу подій, які відбуваються в комп'ютерній системі або мережі, що, в свою чергу, призводить до виконання аналізу можливих ознак інциденту, а також обмеження чи повну заборону несанкціонованого доступу. Всього існує два способи для того, щоб відрізняти нормальну та аномальну поведінку системи, а саме: підхід виявлення аномалій та підхід виявлення неправильного використання.

Виявлення неправильного використання має можливість виявляти послідовності дій як атаку, при наявності збігу між попередніми повними описами серії дій, що мають назву підпис, які виконав зловмисник.

В свою чергу, другий підхід має за мету виявлення шаблонів даних, що не відповідають поведінковим очікуванням. Машинне навчання є основним методом,

що є практично цінним для виявлення аномального втручання в систему. Основоположними принципами підходів до машинного навчання є навчання даних та прогнозування. Машинне навчання має в собі дві категорії, а саме методи обчислювального інтелекту, що включає в себе загальні алгоритми, штучні нейронні мережі (ANN), штучні імунні системи (AIM), нечітку логіку та методи штучного інтелекту, що має в собі KNN, DT, SVM, кластеризацію k-середніх та MLP. Є важливим позначити те, що сфера машинного навчання займається розробкою систем з можливістю автоматичного навчання даних, що включає в себе ідентифікацію шаблонів, які автоматично приховані.

Технології машинного навчання є ефективними відносно покращення рівня виявлення та зменшення кількості та частоти помилкових тривог, а також здійснюють забезпечення зменшення витрат на обчислення та зв'язок. Техніки машинного навчання мають в собі різні фази, а саме контрольоване, напівконтрольоване та неконтрольоване навчання та навчання з підкріпленням (RL).

В контрольованому навчанні алгоритми здатні вивчати уявлення для того, щоб передбачити випадки, що є невідомими, з позначених введених даних. Приклади керованих алгоритмів машинного навчання включають в себе такі алгоритми як Випадковий ліс для вирішення проблем класифікації та регресії і SVM для вирішення питань щодо класифікації. SVM є поширеним рішенням, що використовується в дослідженнях на основі систем виявлення вторгнень через їх практичність обчислень та їх потужну класифікацію.

Алгоритм випадкового лісу є ансамблевим підходом контрольованого навчання, що дуже ефективно справляється з нерівномірними даними, але має проблеми в переобладнанні.

В схемі неконтрольованого навчання структура, що складається з введених даних без позначень визначається алгоритмами. За мету даного алгоритму є створення моделей базових структур даних для прогнозування невідомих даних. Деякі алгоритми неконтрольованого навчання включають в себе методи, які

застосовуються для зменшення функцій, а саме RSA, методи кластеризації та самоорганізаційні карти (SOM). SOM відноситься до штучної нейронної мережі, яка застосовується для зменшення корисного навантаження системи виявлення вторгнень. RSA є алгоритмом, що застосовується для ефективного прискорення вивчення функцій, що не є контрольованими. Тому, алгоритми кластеризації, а саме k-сусідів використовують для виявлення аномалій. Ці алгоритми мають недоліки, а саме суб'єктивність початкових умов, випадки та виробництво високого рівня помилкових спрацьовувань. Використовуючи кореляційні функції можуть маркувати карти, що були створені SOM. Крім цього, SOM часто застосовується для виявлення вторгнень.

Методи навчання з підкріпленням (RL) навчається із навколишнього середовища, що є одним із початкових методів навчання, що застосовуються людьми. Є природним те, що люди починають навчання через взаємодію з навколишнім середовищем. Тому, навчання з підкріпленням має за основу нейронаукові та психологічні аспекти поведінки тварин в поєднанні з механізмами, за допомогою чого агенти мають можливість посилення контролю середовища. Крім того, навчання з підкріпленням гарантує те, що агент здатний мати знання відносно відображення ситуацій для отримання максимальної винагороди. Цей агент не має пам'яті відносно очікуваних подій, але спершу він має дізнатися про події, які приносять максимальну винагороду за допомогою методом спроб та помилок. Ця функція є унікальною та основною характеристикою навчання з підкріпленням. Таким чином, агент на постійній основі навчається на своєму минулому досвіді для того, щоб отримати кращу винагороду. Спостереження систем виявлення вторгнень на основі машинного навчання для безпеки IoT має складну реалізацію звичайних підходів, а саме спостереження системи виявлення вторгнень на основі машинного навчання тому що, вони мають складну структуру технологій, рівнів та стеків протоколів. Тому, є висока необхідність для систем виявлення вторгнень в регулюванні безпечної та надійної мережі між пристроями Інтернету речей. Для розробки системи виявлення вторгнень використовують такі

методи машинного або глибокого навчання як випадковий ліс (RF), штучні нейронні мережі (ANN), самоорганізуюча карта (SOM) та наївний байєсівський алгоритм (NB).

Підходи до виявлення вторгнень мають високі показники помилкового спрацювання. Через те, що методи виявляють будь-які відхилення від базового значення як вторгнення, то відсутність втручання в поведінку, що відрізняється від нормальної, також має позначення як вторгнення, тому результат буде хибно позитивним.

Саме тому, алгоритм на основі правил є хорошим засобом при виявленні атак, що раніше були невидимі. Через це, адміністратор може застосувати превентивні заходи захисту, щоб припинити роботу підозрілої програми та додавання IP-адреси передбачуваного джерела атаки до фільтра брандмауера або можливе тимчасове відключення системи від мережі.

Ізоляційний ліс — це неконтрольований алгоритм виявлення вторгнень, що має за основу використання алгоритму випадкового лісу, щоб виявляти аномалії у наборах даних. Цей алгоритм має за мету розділення або відокремлення точок даних так, щоб кожне спостереження було під ізоляцією відносно інших.

В більшості випадків аномалії знаходяться далеко від кластера точок даних і це дозволяє легше виділяти аномалії в порівнянні із звичайними точками даних.

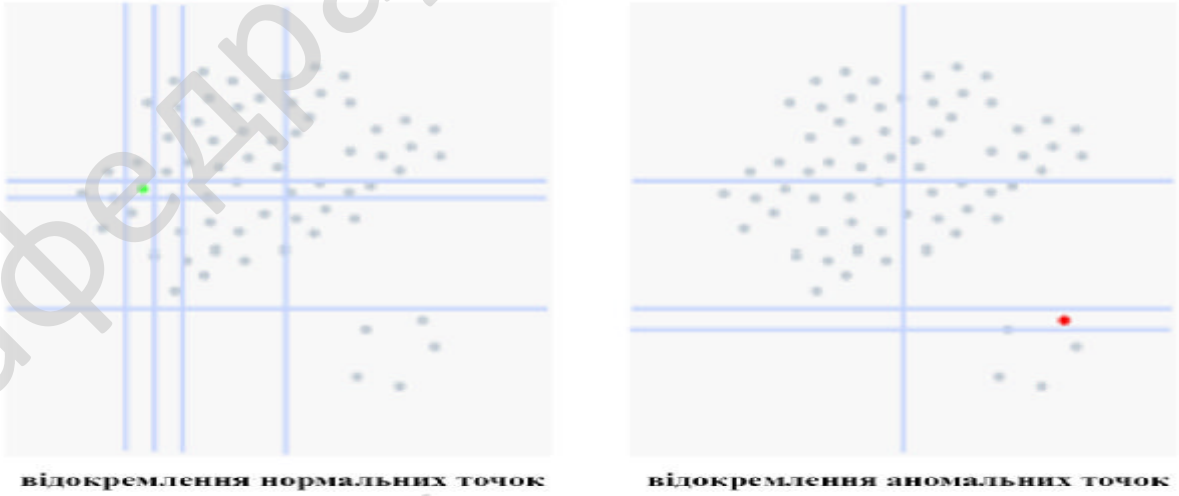

Рис. 3.1 - Розбиття аномальних і звичайних точок

Ці зображення вказують на те, що звичайні точки даних потребують значно більшу кількість розділів, ніж точки даних аномалії.

Оцінка аномалії обчислюється для всіх точок даних: якщо оцінка більша ніж порогове значення, то її можна вважати аномальною.[30]

Логістична регресія виконує оцінку ймовірності події, беручи за основу набір даних незалежних змінних. Так як результатом є ймовірність, то залежна змінна не може виходити за межу проміжку 0 та 1. Логістична регресія застосовує логіт-перетворення до шансів, що означає поділ ймовірності успіху на ймовірність невдачі.

В машинному навчанні логістична регресія відноситься до сімейства керованих моделей машинного навчання. Ця регресія вважається також дискримінаційною моделлю, а саме може розрізняти класи.

Логістична регресія максимізує функцію правдоподібності журналу для визначення бета-коефіцієнтів моделі. Ця функція змінюється в машинному навчанні. У машинному навчанні негативну логарифмічну ймовірність використовують як функцію втрат, що використовує процес градієнтного спуску, для знаходження глобального максимуму.

Логістична регресія має можливість схильності до переобладнання, особливо при наявності в моделі великої кількості змінних предикторів. Регуляризація зазвичай застосовується до штрафування параметрів великими коефіцієнтами, коли модель зазнає страждань від високої розмірності.[31]

Випадковий ліс — це конструкція даних, яка застосована до машинного навчання, що сприяє розроблянню великої кількості дерев випадкових рішень, аналізуючи набори змінних. Алгоритм цього типу допомагає вдосконалювати способи технологій аналізу складних даних.

Цей алгоритм складається із багатьох дерев рішень. “Ліс”, що був згенерований за допомогою алгоритму випадкового лісу, виконує процес навчання за допомогою завантажувального агрегування або пакетування. Пакетування є

ансамблевим метаалгоритмом, що виконує покращення точності алгоритмів машинного навчання.

Алгоритм випадкового лісу задає результат, беручи за основу передбачення дерев рішень. Цей алгоритм виконує прогноз на основі середнього значення виходу з різних дерев рішень. Точність результату алгоритму підвищується разом із збільшенням кількості дерев.

Випадковий ліс забезпечує усунення обмежень алгоритму дерева рішень. Це впливає на збільшення точності і зменшення переобладнання наборів даних. Алгоритм генерує прогнози, без вимог багатьох конфігурацій у пакетах.[35]

Основна відмінність між алгоритмами випадкового лісу і деревом рішень є в тому, що в алгоритмі випадкового лісу розділення вузлів та встановлення кореневих вузлів виконується випадково. Алгоритм випадкового лісу має за основу метод пакетування, що впливає на створення необхідного прогнозу.

Пакетування передбачає використання різних навчальних даних, а не використовуючи лише один зразок. В собі, навчальний набір даних вміщує функції і спостереження, що використовуються для прогнозів. Дерева рішень дають різні результати, в залежності від тих навчальних даних, що передаються алгоритму випадкового лісу. З цими результатами виконують ранжування і вибирається найвищий як кінцевий результат.[34]

Регресією є завдання, що виконує алгоритм випадкового лісу. Ця регресія дотримується правил та концепції звичайної регресії. Значення незалежних та залежних змінних передаються в моделі випадкового лісу.

Ці випадкові регресії можуть бути запущеними в таких мовах програмування як Python, R та SAS. Регресія випадкового лісу - кожне дерево відповідає за створення певного прогнозу. Її результатом є середнє передбачення окремих дерев. Цей результат є суперечленням до випадкової класифікації лісу, результат якої є режим класу дерев рішень.

Не беручи до уваги те, що лінійна регресія та регресія випадкового лісу наслідують однакову концепцію, вони мають різні функції. Функція лінійної регресії має наступний вигляд:

$$y=bx + c, \quad (3.1)$$

де y – залежна змінна, x – незалежна змінна, c — константа, а b – параметр оцінки.

А функція комплексної регресії випадкового лісу більше має схожість із чорною скринею.[35]

Одна із найбільш важливих особливостей цього алгоритму є можливість оброблення наборів даних, які містять безперервні змінні у випадку регресій та категоріальні змінні у випадках класифікацій. Це забезпечує кращі результати для проблем класифікації.

Зазвичай дерева рішень є популярними для завдань машинного навчання. У випадковому лісі будуються набори дерев випадкових рішень для того, щоб ретельніше ізолювати знання від обміну даними з застосованими змінними масивами. В різних видах машинного навчання, випадковий ліс допомагає деталізації технологічних систем і забезпечення більш складного аналізу.

Ці дві методики є дуже корисними при використанні їх в машинному навчанні. Частота невиявлення (UND) та частота помилкових тривог (FAR) є однаково важливими як показники продуктивності моделей машинного навчання в області безпеки. Тому, для перевірки продуктивності моделей Випадкового лісу та Лінійної регресії проводяться виміри частот невиявлення і помилкових тривог. FAR – це відсоток нормального трафіку, який був неправильно класифікований як аномальний. В свою чергу UND є протилежністю до FAR, а саме відсоток трафіку, що являє собою аномалії, які були класифіковані як нормальні. Також використовується загальна похибка та точність. Всі ці 4 пунки зображені в рівняннях нижче.

$$FAR = \frac{\text{хибно позитивний}}{\text{істинно негативний} + \text{хибно позитивний}} \times 100\% \quad (3.1)$$

$$UND = \frac{\text{хибно негативний}}{\text{хибно негативний} + \text{істинно позитивний}} \times 100\% \quad (3.2)$$

$$\text{Загальна помилка} = \frac{\text{хибно позитивний} + \text{хибно негативний}}{\text{кількість зразків}} \times 100\% \quad (3.3)$$

$$\text{Загальна точність} = \frac{\text{істинно позитивний} + \text{істинно негативний}}{\text{кількість зразків}} \times 100\% \quad (3.4)$$

З цього слідує, що тут невірно спрацьовує кількість нормальних пакетів, що визначаються як аномальні. Тому, true negative – це кількість правильно ідентифікованих нормальних пакетів; false negative – це кількість аномальних пакетів, що визначені як нормальні; true positive – це кількість правильно виявлених аномальних пакетів.[33]

Алгоритм kNN – це алгоритм машинного навчання, що використовується для проблем класифікації в data science. Цей алгоритм є не тільки одним із найпростіших, а й дуже широко використовуваний алгоритм.

Фундаментальне припущення в алгоритмі найближчих сусідів полягає в тому, що подібні спостереження знаходяться близько один до одного, а викиди є самотніми спостереженнями, що знаходяться далеко від кластера подібних спостережень.

Не звертаючи увагу на те, що алгоритм найближчих сусідів є контрольованим алгоритмом машинного навчання, коли цей алгоритм виявляє аномалії, то використовується неконтрольований підхід. Це пов’язано з тим, що в процесі не відбувається фактичного навчання, а набір даних не має наперед визначених “аномалій” або “не аномалій”, тому що цей алгоритм базується на порогових значеннях.[36]

Алгоритм (Support Vector Machine) є алгоритмом керованого машинного навчання, що часто використовується в проблемах класифікації. SVM використовують гіперплощини в багатовимірному просторі, для відокремлення

одного класу спостережень від інших. Тому, цей алгоритм часто використовується для рішення завдань багатокласової класифікації.

Але, при цьому, алгоритм SVM частіше використовується в задачах одного класу, де всі дані відносяться до одного класу. В цьому випадку алгоритм виконує “навчання” для того, щоб визначити “нормальність” нових даних, щоб алгоритм мав можливість визначити чи належать вони до групи чи ні. При негативному результаті, нові дані отримують позначення аномальних чи незвичайних.[37]

Штучна нейронна мережа — це керований алгоритм машинного навчання, що бере за основу роботу нервової системи людського мозку. Цей алгоритм вміщує в себе елементи обробки, які ще мають назву нейронні вузли та з’єднання між ними. Ці вузли організовані на вхідному рівні, великої кількості прихованих шарів та вихідному рівні. Алгоритм зворотнього поширення застосовується як техніка навчання штучної нейронної мережі. Головна перевага цього методу полягає у здатності виконання нелінійного моделювання шляхом навчання, використовуючи великі набори даних. Але основна проблема цього навчання в моделі штучної нейронної мережі є в тому, що через її складний характер та в результаті високої витрати часу, це сповільнює процес навчання та досягнення оптимального рішення.

Для того, щоб подолати обмеження штучної нейронної мережі, Гуан-Бінь Хуанг запропонував нову штучну нейронну мережу, що має назву машинне екстремальне навчання. Машинне екстремальне навчання є нейронною мережею прямого зв’язку з одним прихованим шаром, що випадково застосовує вхідні ваги та зміщення прихованого шару не виконуючи налаштування та визначає вихідні ваги за допомогою аналітичного шляху. Взявши за основу екстремальне машинне навчання, Гоцян Лі запропонував мережу швидкого навчання. Мережа швидкого навчання має за основу паралельне з’єднання багаторівневої прямої нейронної мережі та однорівневої прямої нейронної мережі. Мережа швидкого навчання показала чудову стабільність і продуктивність, застосовуючи менше прихованих вузлів та затрачаючи менше часу на роботу.[39]

Мохаммед Хасан Алі звернувся до проблеми систем виявлення вторгнень, запропонувавши модель, що була заснована на мережі швидкого навчання і оптимізації рою частинок, тестування моделі виконувалось за допомогою набору даних KDD Cup'99. Тестування моделі проводилося шляхом порівнянь мережі швидкого навчання із іншими алгоритмами оптимізації. В результаті було отримано, що ця модель перевершує інші моделі мереж швидкого навчання, а саме оптимізація на основі вдосконаленого викладання, генетичний алгоритм та оптимізація гармонічного пошуку. Ці алгоритми також продемонстрували те, що сприяння збільшенню кількості нейронів у прихованому шарі підвищує точність результату навчання моделі. Однак є недолік, він полягає в тому, що цей алгоритм має нижчу точність виявлення для нижчих класів атаки.[40]

Ансамблевий метод — це алгоритм, ключова ідея якого полягає в отриманні користі від різних класифікаторів, за допомогою навчання в ансамблі. Причиною цьому є наявність слабких сторін у кожного класифікатора. Деякі класифікатори вміють розпізнавати певний тип атак, але мають низьку ефективність при інших атаках. Тому, ансамблевий метод має за мету поєднання слабких класифікаторів, навчаючи декілька класифікаторів, після чого слідує утворення більш сильного класифікатора, за допомогою алгоритму голосування.[40]

Яньпін Шень запропонував систему виявлення вторгнень, яка використовує метод ансамблю, беручи за основу класифікатора машинне екстремальне навчання. Для того, щоб оптимізувати запропоновану методологію на етапі ансамблевого скорочення, було використано алгоритм оптимізації BAT. Цю модель було перевірено, використовуючи набір даних KDD Cup'99, Kyoto та NSL-KDD. Результати експерименту вказали на те, що багато моделей екстремального машинного навчання, що були об'єднанні в ансамбль, мають значну перевагу ніж окремі моделі екстремального машинного навчання за продуктивністю. [38]

Сянвей Гао запропонував модель адаптивного ансамблю, за основу береться використання декількох базових класифікаторів, як KNN, DT, RF, DNN (глибока

нейронна мережа), після чого вибирати найкращий, за допомогою адаптивного алгоритму голосування.

Ця методологія була перевірена через проведення експериментів, в яких використовували набори даних NSL-KDD. Експериментальні результати показали ефективність продуктивності, порівнюючи з іншими моделями. Але при цьому, ця методологія не дала задовільних результатів при більш слабких класах атак.

Алгоритми виявлення аномалій є надзвичайно корисними при виявленні випадків шахрайства або зламів, де розподіл цільового класу є дуже незбалансованим. Ці алгоритми також мають в собі за мету подальше покращення продуктивності моделі, використовуючи шляхи видалення аномалій із навчальної вибірки.[40]

3.2 Розробка структурної схеми

На рисунку 3.3 зображена структурна схема системи виявлення вторгнень з використанням алгоритму машинного навчання.

Підписи систем виявлення вторгнень зберігаються в базу даних підпису, а шаблони з даними зберігаються разом із цими збереженими підписами для вторгнень. Висока ефективність забезпечується завдяки перевірці кожного підпису відомих атак. Але, з іншого боку, цей метод не здатний виявляти атаки та вторгнення через відображення сигнатурних шаблонів. Окрім того, в наявності є підтримка величезної бази даних підписів, що в свою чергу порівнюється з пакетами даних для виявлення можливих вторгнень, але це є дуже ресурсозатратним. Виявлення порушення на основі аномалії має за основу ідею чіткого визначення профілю нормальної діяльності. Будь-які відхилення від норми профілю помічається як аномальна активність. Переваги у виявленні порушення на основі аномалії полягає у тому, що він здатний виявляти нові та невідомі атаки та наявність індивідуального характеру нормального профілю активності для різних програм та мереж. Але, в свою чергу, одним із основних недоліків виявлення

порушень на основі аномалії полягає в тому, що він має високий FAR, через це є дуже важкою задачею знайти межу між нормальним та аномальним профілем для вторгнення. Через популярність парадигми Інтернету речей та завдяки розвитку мережевих технологій привело до підвищеної уваги до використання пристроїв IoT. Одна із найважливіших технологій, які використовуються в розробках мереж IoT — це WSN, що має в собі набір сенсорних вузлів для виконання збору інформації. Більша кількість інформації, що є важливою збираються сенсорними пристроями IoT та виконують передачу через мережу Інтернет. Але це, та наявність складної структури WSN негативно впливає на безпеку мережі IoT. Тому системи виявлення вторгнень є визнаним одним із ефективних механізмів для забезпечення безпеки WSN та IoT.

Watchdogs є мережевими вузлами, що мають за призначення моніторинг та спостереження мережевого трафіку сусідніх вузлів. Потім, ці вузли приймають рішення відносно несправних вузлів, базуючись на певний набір правил. Є багато пропозицій рішень для виявлення вторгнень та аномалій, використовуючи сторожеві системи у домені WSN39, IoT та AdHoc. Моделі довіри — інстумент, що застосовується для збільшення продуктивності систем виявлення вторгнень. Система виявлення вторгнень на основі цієї моделі виконує перевірку надійності своїх вузлів, для ідентифікації зловмисних вузлів, а також постійно тримаючи мережевий трафік під контролем, для виявлення аномальної поведінки. Різні реалізації систем виявлення вторгнень, що використовують моделі довіри мають за основу сторожові тайми, моделі довіри на основі теорії ігор та байєсівській моделі довіри. В IoT схема довірчого управління може застосовуватися розподіленим способ, що в свою чергу, зменшує обчислювальні накладні витрати вузлів датчиків з обмеженими ресурсами. Також, для ефективного проектування систем виявлення вторгнень, використовують теорію ігор. Теорія ігор — це математична концепція, що застосовується для моделювання стратегічної взаємодії між гравцями, використовуючи опис гри. Кожна гра має в собі набори гравців, в свою чергу, кожен гравець має набір правил або стратегій та план дій і виграш за кожну дію в

певній грі. Кожна гра може бути як кооперативною так і ні, залежно від взаємодії суб'єктів у конкурентному або кооперативному режимах. З точки зору системи виявлення вторгнень для Інтернету речей та WSN моделюється між захисниками та зловмисниками або шляхом їх взаємодії, чи стратегії прогнозування зловмисника.

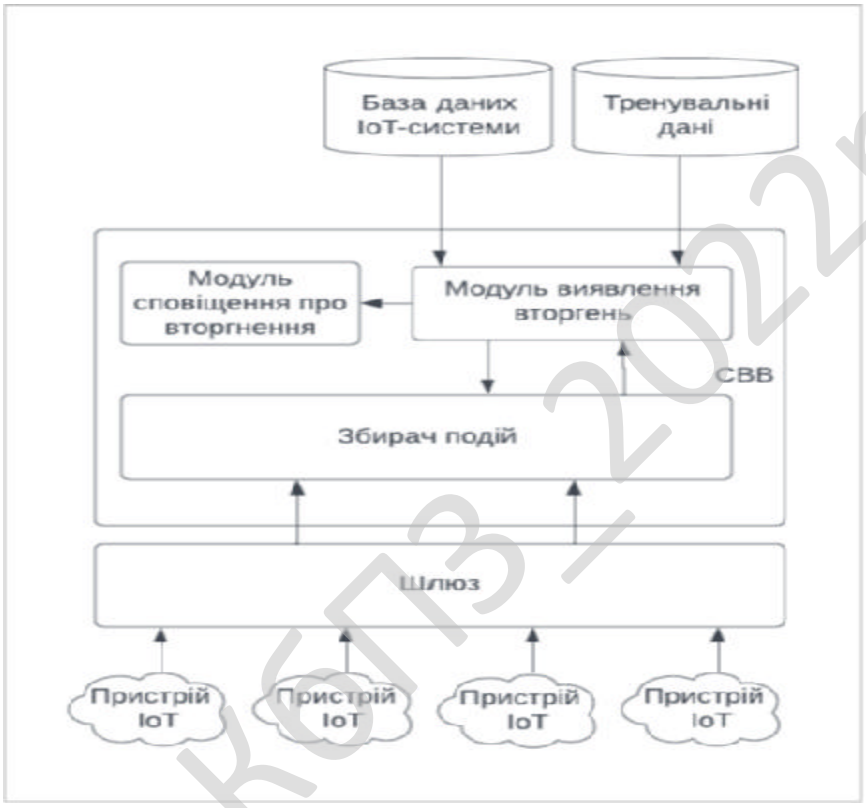

Рис. 3.2 - Структурна схема системи

Збирач подій відповідає за збір та запис усіх подій, що здійснюються пристроями IoT для побудови поточної поведінки.

Модуль виявлення аномалій відповідає за аналіз та виявлення вторгнень. Цей модуль є основним компонентом системи виявлення вторгнень.

Модуль сповіщення про вторгнення відповідає за блокування користувача та завершення його сесії, при виявленні атаки та надсилає сповіщення адміністратору для виконання відповідних дій.

Разом із зростанням кількості додатків та користувачів в мережі безпека стала однією із найважливіших проблем мережевих систем. На фізичному рівні

проблеми включають в себе фізичні пошкодження, збої пристрою та обмеження живлення. А в свою чергу до проблем мережевого рівня відносяться сніфери, відмови в обслуговуванні, незаконний доступ та атаки на шлюз. Багато пристроїв Інтернету речей мають лише систему самозахисту і через це є дуже вразливими до вторгнень. Проблеми фізичних загроз та автентифікації є початковими перешкодами, що повинні бути подолані системою IoT. Між пристроями IoT та шлюзами мережевого рівня є проблема конфіденційності. Також не варто забувати про проблему, яка пов'язана із цілісністю даних, які передаються між програмами та службами. Ці проблеми виникають тоді, коли на мережеву систему було здійснено атаку спуфінгу або шумом. DDoS, DoS та зондувальні напади є атаками, що можуть поставити під загрозу служби IoT та систему. Конфіденційність є надзвичайно важливим аспектом безпеки в системах IoT. Різні компоненти IoT застосовують багато різних методів ідентифікації елементів. Тому, в результаті кожна річ має унікальний ідентифікаційний тег, що містить в собі персональні дані, дані про місцезнаходження та переміщення, що є критично важливою інформацією, яка повинна бути захищена.

3.3 Розробка функціональної схеми

На рисунку 3.4 зображена функціональна схема системи. Нижче ця система буде розглянута більш детально.

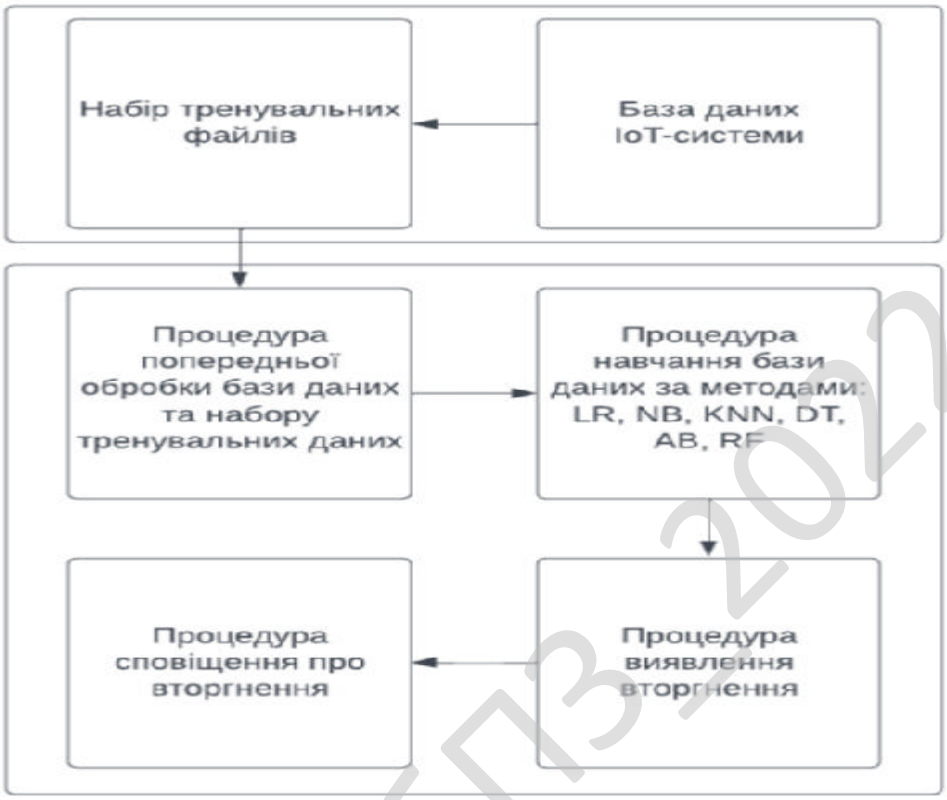

Рис. 3.3 - Функціональна схема системи

Процедура попередньої обробки бази даних та набору тренувальних даних. Призначена для перетворення вхідних даних у дані, що є необхідними для машинного навчання. Машинне навчання включає в себе перетворення, стандартизацію, бінаризацію та нормалізацію. Набори даних проходять попередню обробку для розгляду проблеми двійкової класифікації, в якій розглядаються лише дві мітки, тобто звичайний трафік та аномальні вторгнення, а перетворення та нормалізація даних застосовується до набору даних.

Завдяки нормалізації навчання стає менш чутливим у функціональному масштабі та забезпечує гарантію того, що проблема конвергенції не має великої дисперсії, яка робить можливою оптимізацію. Після операції попередньої обробки результатом є утворення набору даних.

Процедура навчання бази даних за методами: LR, NB, KNN, DT, AB, RF. Призначена для вибору найбільш значущих об'єктів із заданого набору даних. Завдяки цьому можна підвищити ефективність зберігання, зменшити витрати для обчислення та покращення продуктивності моделі автоматичного завдання. Є декілька способів для вибору функцій, що згруповані в три категорії: методи фільтрації, а саме вибір функції за допомогою кореляційної матриці. До кожної функції застосовується оцінювання на основі статистичних розрахунків; тоді проводиться вибір лише тих атрибутів, де кореляція перевищує порогове значення. Wrapper Methods, що виконує пошук найбільш оптимальної комбінації функцій, виконуючи оцінку точності моделі. А це означає, що здійснюється передача функції до вибраного алгоритму машинного навчання та на основі продуктивності моделі додаються або видаляються функції.

База даних. Було використано базу даних KDD Cup 1999, яка імітує мережу військово-повітряних сил США. Ця база даних була розроблена MIT Lincoln Labs та надає стандартний набір даних, що був створений на основі симуляцій у військових мережевих середовищах та шляхом охоплення різноманітних вторгнень. Цей набір включає в себе три незалежні набори: "повна KDD", "10 % KDD" та "виправлена KDD". Було використано "10 % KDD" та "виправлений KDD" як набори даних для навчання та тестування відповідно. З'єднанням в наборі даних KDD Cup 99 є послідовність TCP-пакетів, що має в собі 42 функції, які в свою чергу позначені як звичайні або атакуючі.[44]

Процедура виявлення вторгнення. Для оцінки продуктивності системи виявлення вторгнень, багато показників обчислюються за допомогою значень у матриці помилок. Ці значення мають наступний опис: True Positive(TP): кількість записів, що були правильно віднесені до класу normal. True Negative(TN): кількість записів, які були коректно віднесені до класу attack. False Positive(FP): кількість normal записів, що були неправильно віднесені до класу attack. False Negative(FN): кількість записів attack, які були некоректно віднесені до класу normal. Тому, виходячи з вище наведених значень, оцінки задаються наступними формулами:

$$\text{TP rate} = \frac{TP}{TP + FN} \quad (3.5)$$

$$\text{TN rate} = \frac{TN}{TN + FP} \quad (3.6)$$

$$\text{FP rate} = \frac{FP}{FP + TN} \quad (3.7)$$

$$\text{FN rate} = \frac{FN}{FN + TP} \quad (3.8)$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.9)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (3.10)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3.11)$$

$$\text{F1 Score} = 2 * \frac{Precision \cdot Recall}{Precision + Recall} \quad (3.12)$$

Після виконання обчислень показників для різних моделей навчання, вони записуються як звіт у вигляді показників. Ці показники є результатами навчання моделей та їх точності.

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання магістерського проектування наведена на рисунку 3.4. З цього рисунку можна

побачити, що робота програмного продукту починається з запуску проекту початку/кінця програми.

Цей процес взаємодіє з процесом відкриття головного вікна ПЗ системи IoT.

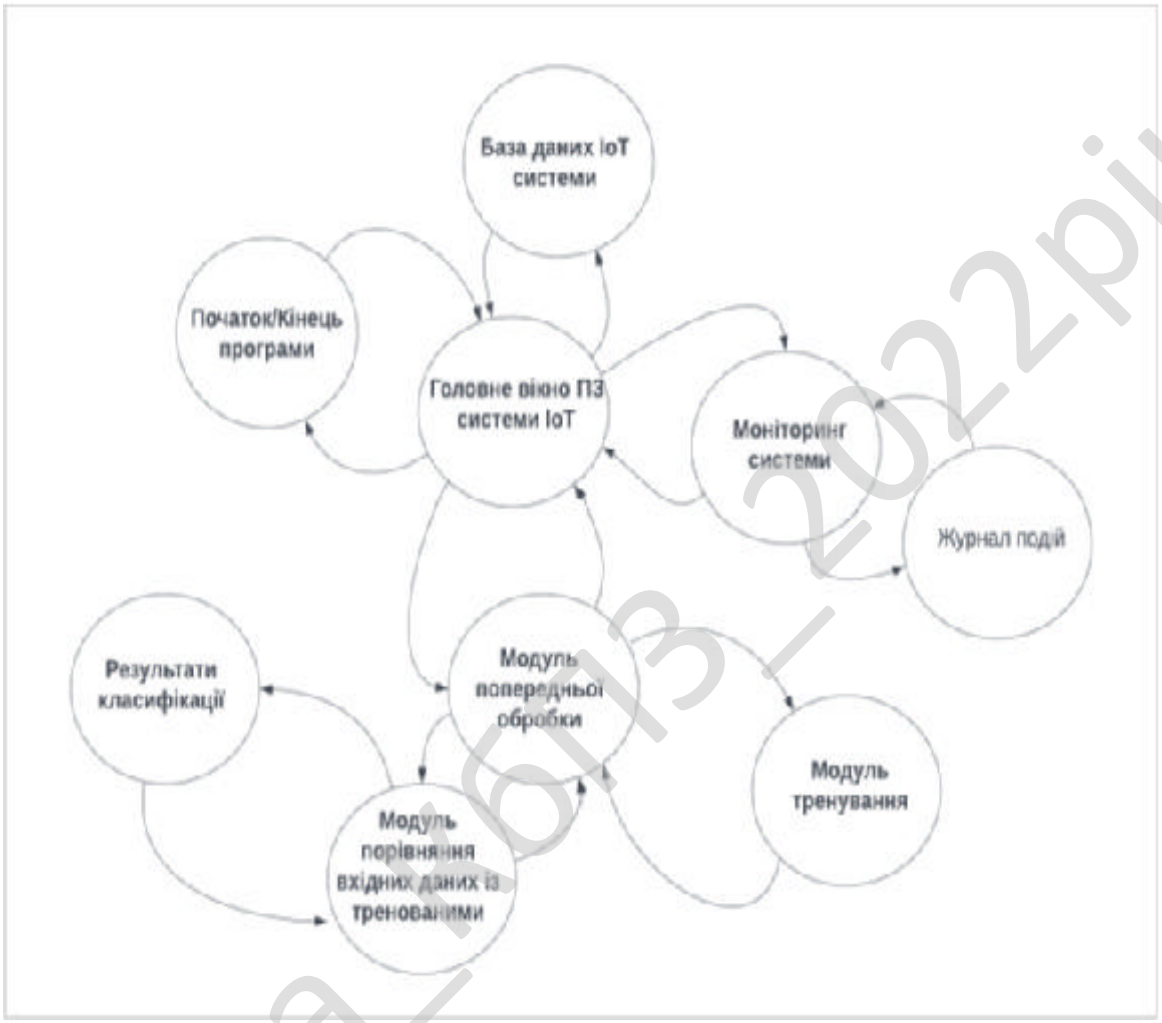

Рис. 3.4 — Діаграма процесів системи

Процес відкриття головного вікна ПЗ системи IoT взаємодіє з наступними процесами:

- Процесом моніторингу системи.

- Процесом надання доступу до бази даних IoT системи.

Процес моніторингу процесу взаємодіє з процесом відображення журналу подій.

Процес попередньої обробки взаємодіє з наступними процесами:

- Процесом тренування бази даних.

- Процесом порівняння тренованих моделей з вхідними даними.

- Процес виводу результату класифікації.

На цьому програмний продукт закінчує свою роботу.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів можна перейти до опису блок-схем основної програми, та підпрограми, що використовуються для реалізації системи.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 наведено блок-схему основної програми. Її робота складається з виконання наступних кроків:

- Виведення основного вікна програми.
- Виконується підключення до бази даних KDD Cup’99.
- Виконується навчання тренувальних даних за заданими правилами.
- При наявності аномалій в системі, виконується внесення їх до списку.
- Складається загальний звіт перевірки системи.
- Виведення результату класифікації.

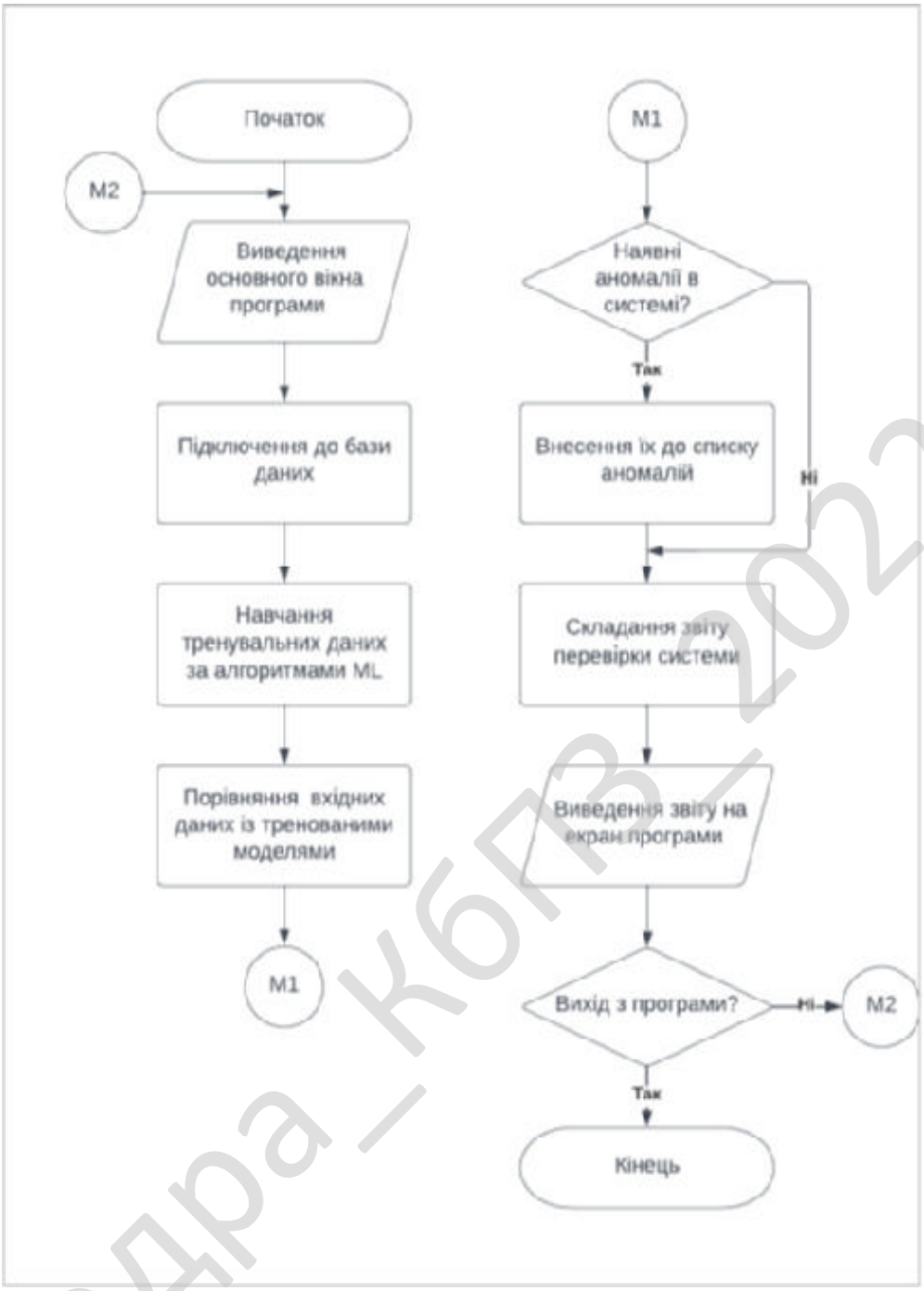

Рис. 4.1 - Блок-схема роботи основної програми

Блок-схема роботи підпрограми попередньої обробки та порівняння зображена на рисунку 4.2.

Вона складається з наступних кроків:

- Завантаження моделі навчання з файлу.
- Виконується підключення до бази даних KDD Cup’99.

- Навчання даних за наступними моделями навчання: логістична регресія (LR), байєсівський алгоритм (NB), knn-сусідів (KNN), дерево вибору (DT), AdaBoost (AB), випадковий ліс (RF).
- Вивід результату навчання моделей у вигляді наступних показників: accuracy, precision, recall, f1 score.

Рис. 4.2 - Блок-схема підпрограми

Навчання даних за моделями.

Логістична регресія (LR):

```
print("----------------------------------------LR-------------------------
------")
model = LogisticRegression()
model.fit(traindata, trainlabel)

# створення прогнозів
expected = testlabel
np.savetxt('classical/expected.txt', expected, fmt='%01d')
predicted = model.predict(testdata)
proba = model.predict_proba(testdata)

np.savetxt('classical/predictedlabelLR.txt', predicted, fmt='%01d')
np.savetxt('classical/predictedprobaLR.txt', proba)

y_train1 = expected
y_pred = predicted
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")

print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)
```

Байєсівський метод (NB):

```
# Підбір Naive Bayes модель до бази даних
print("----------------------------------------NB-------------------------
------")
model = GaussianNB()
model.fit(traindata, trainlabel)
print(model)
# створення прогнозів
expected = testlabel
predicted = model.predict(testdata)
proba = model.predict_proba(testdata)

np.savetxt('classical/predictedlabelNB.txt', predicted, fmt='%01d')
np.savetxt('classical/predictedprobaNB.txt', proba)

y_train1 = expected
y_pred = predicted
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")

print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
```

```
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)
```

knn-сусідів (KNN):

```
# підібрати модель k-найближчого сусіда до даних
print("-----------------------------------------KNN-------------------------
-------")
model = KNeighborsClassifier()
model.fit(traindata, trainlabel)
print(model)
# створення прогнозів
expected = testlabel
predicted = model.predict(testdata)
proba = model.predict_proba(testdata)


np.savetxt('classical/predictedlabelKNN.txt', predicted, fmt='%01d')
np.savetxt('classical/predictedprobaKNN.txt', proba)
# Підсумування відповідності моделі

y_train1 = expected
y_pred = predicted
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")


print("-----------------------------------------------")
print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)
```

Дерево рішень (DT):

```
print("-----------------------------------------DT--------------------------
------")

model = DecisionTreeClassifier()
model.fit(traindata, trainlabel)
print(model)
# створення прогнозів
expected = testlabel
predicted = model.predict(testdata)
proba = model.predict_proba(testdata)

np.savetxt('classical/predictedlabelDT.txt', predicted, fmt='%01d')
np.savetxt('classical/predictedprobaDT.txt', proba)
# Підсумування відповідності моделі


y_train1 = expected
y_pred = predicted
```

```
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")


print("-----------------------------------------------")
print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)
```

AdaBoost (AB):

```
print("-----------------------------------------Adaboost--------------------
------------")

model = AdaBoostClassifier(n_estimators=100)
model.fit(traindata, trainlabel)

# створення прогнозів
expected = testlabel
predicted = model.predict(testdata)
proba = model.predict_proba(testdata)

np.savetxt('classical/predictedlabelAB.txt', predicted, fmt='%01d')
np.savetxt('classical/predictedprobaAB.txt', proba)
# Підсумування відповідності моделі

y_train1 = expected
y_pred = predicted
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")


print("-----------------------------------------------")
print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)
```

Випадковий ліс (RF):

```
model = RandomForestClassifier(n_estimators=100)
model = model.fit(traindata, trainlabel)

expected = testlabel
predicted = model.predict(testdata)
proba = model.predict_proba(testdata)
np.savetxt('classical/predictedlabelRF.txt', predicted, fmt='%01d')
np.savetxt('classical/predictedprobaRF.txt', proba)
```

```
      print("-------------------------------------RF-----------------------------
--------")

      y_train1 = expected
      y_pred = predicted
      accuracy = accuracy_score(y_train1, y_pred)
      recall = recall_score(y_train1, y_pred , average="binary")
      precision = precision_score(y_train1, y_pred , average="binary")
      f1 = f1_score(y_train1, y_pred, average="binary")


      print("----------------------------------------------")
      print("accuracy")
      print("%.3f" %accuracy)
      print("precision")
      print("%.3f" %precision)
      print("racall")
      print("%.3f" %recall)
      print("f1score")
      print("%.3f" %f1)
```

Результати класифікації моделей:

```
-------------------------------------LR---------------------------------
       accuracy
       0.846
       precision
       0.988
       racall
       0.819
       f1score
       0.896
       ----------------------------------------NB---------------------------------
       GaussianNB()
       accuracy
       0.929
       precision
       0.988
       racall
       0.923
       f1score
       0.955
       ----------------------------------------KNN----------------------------------
       KNeighborsClassifier()
       -----------------------------------------
       accuracy
       0.929
       precision
       0.998
       racall
       0.913
       f1score
       0.954
       ----------------------------------------DT---------------------------------
       DecisionTreeClassifier()
       -----------------------------------------
       accuracy
       0.931
       precision
```

```
        0.999
        racall
        0.916
        f1score
        0.955
        ----------------------------------------Adaboost-----------------------------
        ----
        -------------------------------------------
        accuracy
        0.925
        precision
        0.995
        racall
        0.911
        f1score
        0.951
        ---------------------------------------RF-------------------------------------
        -------------------------------------------
        accuracy
        0.927
        precision
        0.999
        racall
        0.910
        f1score
0.952
```

Після виконання класифікації моделей на нормальні та аномальні можна зробити висновок, що кращими алгоритмами для виявлення втручань є випадковий ліс та алгоритм дерева рішень. Ці алгоритми є не тільки швидкими, але й дуже точними.

Найменш точним алгоритмом є логістична регресія, не дивлячись на його швидкість.

В порівнянні з ансамблевими методами, алгоритм дерева рішень є більш ефективним. На це вказують результати алгоритму дерева рішень і ансамблевого методу випадкового лісу, в яких дерево рішень має кращі показники: показник accuracy більший на 0,04; показник recall більший на 0,06; показник f1score більший на 0,03. Але вони мають однаковий показник precision, він становить 0,999.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати протокол SSH. SSH – це протокол віддаленого адміністрування, що був розроблений для того, щоб здійснювати віддалене керування операційними системами та тунелювання TCP-з'єднання.[46] При використанні цього протоколу є допустимим використанням RSA алгоритму шифрування, що дає можливість безпечно працювати в більшості незахищених середовищ, наприклад: здійснювати передачу шифрованими каналами будь-яких типів даних, працювати з ПК через командну оболонку.[47]

Протокол SSH використовує клієнт-серверну модель для аутентифікації віддалених систем та забезпечення шифрування даних, які здійснюють обмін в рамках віддаленого доступу.

Для роботи цього протоколу зазвичай використовується TCP-22 порт: на ньому хост-сервер очікує вхідне підключення і після проведення процедури аутентифікації та отримання команди організується запуск клієнта, відкриваючи обрану оболонку. Якщо є необхідність, клієнт може змінювати порт, який використовується на даний момент.[48]

Щоб створити SSH підключення, клієнт має здійснити ініціацію з'єднання з сервером, що забезпечує захищене з'єднання та підтвердження свого ідентифікатора. SSH — хороший вибір для додатків IoT, тому що він є надійним та безпечним. Його застосовують для передачі даних між пристроями або для дистанційного керування пристроями. Також SSH є безпечним засобом віддаленого доступу до пристроїв IoT в мережі. Це є важливим при розгортанні IoT, де пристрої можуть бути розташовані в небезпечних чи недоступних місцях.[45],[49]

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

У розробленій системі виявлення вторгнень в промисловій IoT-інфраструктурі, існують наступні користувальницькі меню:

- Файл.
- Редагування.
- IDS.
- Параметри.
- Довідка.

Роботу програмного забезпечення зображено на рисунках 5.1 та 5.2.

Розроблена програма має дуже простий та зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити.

Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в IT рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності. Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім модулями (сервісами).

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

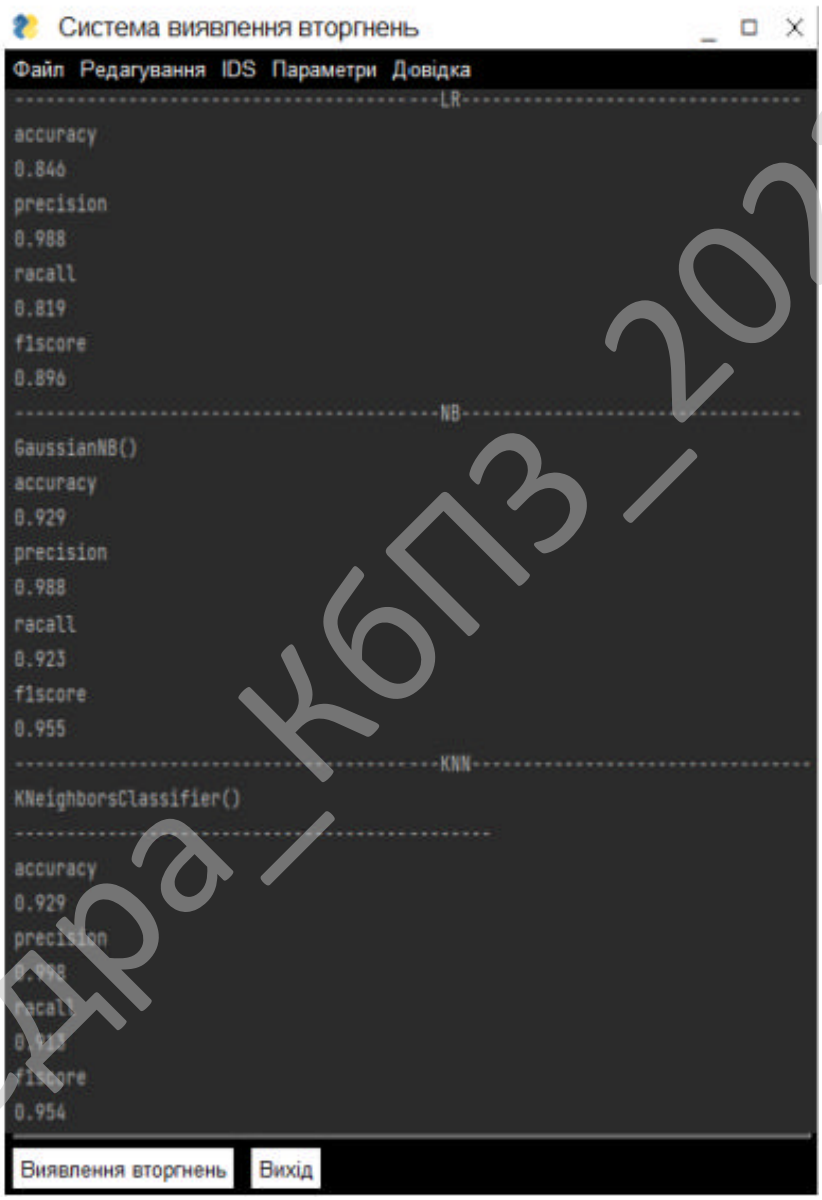

Рис. 5.1 - Головне вікно програми (методи LR, NB, kNN-сусідів)

Рис. 5.2 - Головне вікно програми (методи DT, AB, RF)

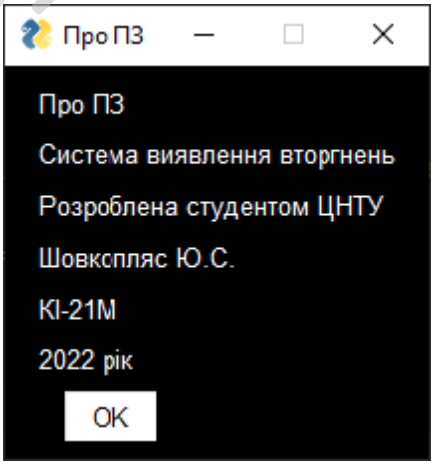

Рис. 5.3 - Вкладка “Довідка”

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи виявлення вторгнень в промислову IoT-інфраструктуру з використанням алгоритму машинного навчання.

Метою розробки є дослідження та програмна реалізація системи виявлення вторгнень в промислову IoT-інфраструктуру з використанням алгоритму машинного навчання.

Об’єктом дослідження є методи системи виявлення вторгнень в промислову IoT-інфраструктуру з використанням алгоритму машинного навчання.

Методи дослідження базуються на методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань обумовлених цілями дослідження, отриманий наступний результат:

- Розроблено вітчизняний продукт системи виявлення вторгнень в промислову IoT-інфраструктуру, який має більш широкі можливості, на відміну від існуючих аналогів. Ці можливості полягають у застосуванні декількох алгоритмів навчання моделі даних та їх порівняння один із одним, що зменшує вірогідність помилкової класифікації подій.

- Удосконалено метод забезпечення безпеки системи промислової IoT-інфраструктури завдяки застосуванню машинного навчання для виявлення вторгнень. Реалізація зазначеного метода дозволяє зменшити завантаженість ресурсів комп’ютерної системи за рахунок спрямування безпечних неоднорідних інформаційних потоків на безпечні та менш завантажені ресурси.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 24 днів (один місяць).

В магістерській роботі було проведене дослідження та виконана програмна реалізація системи виявлення вторгнень в IoT-інфраструктуру.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп'ютері баз даних;

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	50 (ост. цифра № зал $*10^1$)
3. Запланований термін розробки, днів	Fpq	24 (1 місяць)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Г

Продовження таблиці 7.1

1	2	3
6. Складність алгоритму (1, 2, 3)		
7. Кількість макетів вхідної інформації	–	6
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	6
10. Попередній досвід (1-6)	–	1
11. Гнучкість проекту ПП (1-6)	–	2
12. Детальність проекту ПП (1-6)	–	1
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	3
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	5
17. Складність кінцевого програмного продукту (1-6)	–	3
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	2
20. Вимоги до швидкодії ПП (1-6)	–	3
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	4
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	3
25. Постійність складу команди розробників (1-6)	–	1
26. Досвід розробки додатків (1-6)	–	1
27. Досвід роботи з обчислювальною платформою (1-6)	–	1

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	3
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	40000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	37
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	30
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначається по типовим нормам часу, що були проведені в додатках МВ. Стадія РП є найбільш трудомісткою та тривалою, що вносить значний вплив на стан інших стадій проекту.

Трудомісткість розробки ПЗ для стадії РП визначається наступним чином.

Проводиться обчислення номінальних трудовитрат, люд-міс.:

$$T_{ном} = A \, \text{Size}^B, \qquad (7.1)$$

де: А — коефіцієнт Боема, А= 2,45;

Size – загальний об’єм відлагодженого програмного коду, тис. рядків;

В – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001 \Sigma W_i, \quad (7.2)$$

де: W_i – сумарне значення п’яти показників, що забезпечують відображення особливості розробки проекту програмного продукту та колективу розробників.

$$B = 1{,}01+0{,}001*(4{,}05+4{,}86+4{,}22+3{,}95+2{,}73) = 1{,}029$$

$$T_{ном} = 2{,}45 \cdot 0{,}6^{1{,}029} = 1{,}4 \text{ люд-міс.}$$

Визначаються уточнені трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном} \Pi V_j, \quad (7.3)$$

де: Πv_j – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 1{,}4 \cdot (1 \cdot 1{,}19 \cdot 1 \cdot 0{,}91 \cdot 0{,}95 \cdot 1 \cdot 1 \cdot 1{,}15 \cdot 1{,}22 \cdot 1 \cdot 1{,}24 \cdot 1{,}22 \cdot 1{,}25 \cdot 1{,}10 \cdot 1 \cdot 1 \cdot 1{,}10) = 3{,}7 \text{ люд-міс.}$$

Дані коефіцієнти дають можливість диференційовано оцінити результати роботи програмістів, беручи до уваги швидкодію програмного забезпечення, використання обчислювальних платформ та інструментів розробки, взаємодію серверів, вимоги до об’ємів баз даних і т.д.

Визначення підсумкових трудовитрат по стадії робочого проекту, люд-дні вичисляється наступним чином:

$$T_{РП} = 0{,}3 \, CT_{уточн}^{0{,}33+0{,}2 \cdot (B-1{,}01)} \cdot S, \quad (7.4)$$

де: С — визначений емпірично коефіцієнт, запропонований авторами методики;

S – коефіцієнт стиснення графіка робіт %, що дозволяє коректувати термін розробки ПЗ встановленим вимогам. Вибірка прозводиться в межах від 25 до 350%.

$$T_{РП} = 0{,}3 \cdot 2{,}75 \cdot 3{,}7^{0{,}33+0{,}2(1{,}029-1{,}01)} \cdot 50 = 63 \text{ люд-день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводяться відносно таблиці 7.2

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	10	Д5
Ескізний проект	15	Д6
Технічний проект	20	Д7
Робочий проект	63	Ф 7.1-7.4
Впровадження	20	Д13
Всього	128	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \qquad (7.5)$$

де: F_{pq} – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$Ч = \frac{128 \cdot 1}{24 - 3} = 6{,}09$ ставки.

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначення затрат часу на виконання профілактичних робіт по обслуговуванню обладнання за період обробки. Результати зводяться до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	360	6	2160	36
Монітор	180	6	1080	18
Клавіатура	120	6	720	12
Маніпулятор «мишка»	45	6	270	4
Принтер матричний	180	1	180	3
Принтер лазерний	360	1	360	6
Принтер струминний	270	1	270	4
Сканер	180	1	180	3
Концентратор–маршрутизатор	180	2	360	6
Кабельні господарства ЛВС на 1 м. п.	2,5	70	175	3
Кабельне господарство електромережі	48	50	2400	40
Копіювальний апарат	300	1	300	5
Усього за рік:			$З_ч$	140

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків не повинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_{ч} \cdot n_{міс}}{1,2}, \quad (7.6)$$

$$Ф_{др}^{с} = \frac{140 \cdot 1}{1,2} = 116,6 \text{ год.}$$

Необхідна кількість ставок штатного персоналу сектора ТО визначається наступним чином:

$$Ч_{ел} = \frac{Ф_{др}^{с}}{F_{др} \cdot T_{зм}}, \quad (7.7)$$

$Ч_{ел} = 116,6/(24 \cdot 8) = 1$ ставки

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп’ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2019, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN, PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	2	0,5
	Підтримка постійних клієнтів	1	
	Оформлення договорів, ведення тендерів	0,5	
	Контроль взаєморозрахунків з постачальниками	0,5	
Всього		4	
Спеціаліст з кібербезпеки	Спостереження за коректним функціонуванням системи	1	0,5
	Перевірка наявності вторгнень в систему	1	
	Оформлення звітів відносно вторгнень в систему	1	
Всього		3	

Був складений штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	14000	14000
Продакт-менеджер	0,5	10000	5000
Інженер-програміст	6,09	8200	49938
Інженер-електронщик	1	8000	8000
Інженер-системотехнік	0,25	8000	2000
Адміністратор мережі	0,5	8000	4000
Системний програміст	0,25	8000	2000
Всього за період розробки	$R_{сп}$ =9,59	-	$Ф_{роб}$ =84938

Середньоденна зарплата одного виконавця розраховується наступним чином:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де $Ф_{роб}$ — загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{84938}{9{,}59 \cdot 24} = 369 \text{ грн}$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Береться 7 робочих місць;

$S_у$ – питома площа на одне робоче місце, м^2;

$Ц_{пл}$ — вартість одного квадратного метра площі, грн.

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» (м. Кіровоград, вул. Глинки 16) ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 250…1600у.о./м^2. Враховуючи те, що курс у.о. складає 30, що приймається для розрахунку вартості одного метра квадратного рівною 7000 грн./м^2. На кожне робоче місце у середньому потрібно 8м^2. З урахуванням цього:

$Б_{уд} = 7 \cdot 8 \cdot 7000 = 392000$ грн.

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку складає: 39200 грн.

Балансова вартість інвентарю розраховується за нормою 3500грн на робоче місце. Це означає, що:

$$I_{нв} = R_{сп}^{1} \cdot Ц_{м}, \qquad (7.10)$$

де $Ц_м$ — ціна меблів для одного робочого місця, грн.

$I_{нв} = 7 \cdot 3500 = 24500$ грн.

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми Brain за 28.10.22 – джерело https://brain.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова цін
Персональний комп’ютер		24073
Системний блок	VINGA ADVANCED B0166 (R3M8INT.B0166)	24073
Процесор	AMD Ryzen 3, 2100GE, 4 ядра, 4 потоки, Частота процесора, 3.2 ГГц, Частота в Boost, 3.6 ГГц	-
Системна плата	MB AM4, Чіпсет AMD A320, 1 x Headphone, 4 x USB 3.0, 3 x Audio, 1 x Microphone, 4 x USB 2.0, 2 x PS/2, 1 x HDMI, 1 x VGA, 1 x RJ45, Realtek ALC887, 10/100/1000 Мбіт/с	-
Відеокарта	Вбудована AMD Radeon Vega 8	-
Жорсткий диск	SSD 120 GB	-
Оперативна пам’ять	8 ГБ DIMM, DDR4-2666 MHz, PC4-21300	-
DVD-привод	Не комплектується	-
Корпус	ATX Vinga CS108B, PSU 350W(FSP Brand: ATX-400PNR, 12cm), black, (front bezel – black+light silver; body material – 0.6mm), 80mm fan (rear), 2xUSB2.0/AUDIO/MIC, Air Duct, Tool-less chassis design	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова цін
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-B, int. 3.5", 1*USB2.0+AUDIO+1394, multi: All Type Cards, black	420
інше	Клавіатура, мишка	Подару-ног
Монітор	PHILIPS 223V5LSB2/10, 21.5", TN, 1920 x 1080, 16:9, WLED, матове покриття, 5мс	4800
Принтер лазерний	Canon i-SENSYS LBP-223dw	10699
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5870
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	10499

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	7	24073	16420,5	184931
Принтер лаз.	1	10699	540	11239
Принтер струм.	1	5870	550	6420
Копіюв. апарат	1	10499	596,5	11595,5
Всього	–	–	–	214185,5

Витрати на транспорт, монтаж та випробування прийняті в межах до 10% від оптової ціни.

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	682240	-	-
2. Передавальні пристрої	68224	-	-
Всього по групі	750464	5	37523,2
Група 4			
3. Обчислювальна техніка	199177	-	-
Всього по групі	199177	50	99588,5
4. Нематеріальні активи	70000	50	35000
Група 5, 6			
5. Вимірювальні пристрої	9031	25	2257,75
6. Транспортні засоби	143000	20	28600
7. Господарський інвентар	45500	25	11375
Всього по групі	197531	-	42232,75
Разом	$К_р$ = 1217172		$А_р$ =214344,45

Примітка: вартість автомобіля Sens (Standard+) взята по даним з автосалону «Кіровоград-Авто», джерело http://kirovograd-avto.ukravto.ua /catalog/tm-9/model-80/description, складає 143000 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначається основна зарплата виконавців наступним чином:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де $N_е$ – кількість екземплярів програм, шт.

$З_о = 369 \cdot 128/50 = 944$ грн.

Визначається додаткова зарплата на рівні 10% наступним чином:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$З_д = 944 \cdot 10 \cdot 0{,}01 = 94{,}4$ грн.

Відрахування на соціальні потреби за нормативом $Н_с$ = 37% від суми основної та додаткової зарплати:

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де $Н_с$ – відрахування на соціальні потреби, %.

$С_{оц} = 0{,}01 \cdot 37(929+92{,}9) = 384$ грн.

Визначення загальногосподарських витрат за нормативом $Н_г$ = 15% від основної зарплати:

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де: $Н_г$ — загальногосподарські витрати, %.

$Г_{осп} = 944 \cdot 15 \cdot 0{,}01 = 141{,}6$ грн.

Визначення витрат на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_м = (З_{м1} + З_{м2} + З_{м3}) / N_в, \quad (7.15)$$

де: $З_{м1}$ — вартість паперу, грн;

$З_{м2}$ — вартість запам’ятовуючих пристроїв, грн;

$З_{м2}$ — вартість фарби, картріджей, тонеру, грн;

$N_в$ — кількість екземплярів програм, шт.

Згідно виданих викладачем норм приймаємо 0,5 пачки паперу на місяць розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$ = 170 грн., визначаємо вартість паперу за період розробки $N_м$ = 1 міс:

$$З_{м1} = Ц_п \cdot N_м \quad (7.16)$$

$З_{м1}$ = 170·1·0,5 = 85грн.

Згідно виданих викладачем норм до вартості запам'ятовуючих пристроїв входить вартість CD дисків в кількості, що дорівнює 30 екземплярів програм та одного DVD диска для збереження резервної копії програми:

$$З_{м2} = \sum Ц_д, \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 3 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 3 грн./шт.

$З_{м2}$ = 30·3+3 = 93 грн.

Згідно виданих викладачем норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{м3} = \sum Ц_з, \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: картридж для CANON LBP-3010 Black Canon 712 – 2672 грн.; картридж для EPSON STYLUS PHOTO R390 – 789 грн.; картридж для CANON IR-1022A – LJ Q2612A Cart. HP LJ 1010/1012/1015/3015/3020/3030 (2500 стр.) – 688 грн.

$З_{м3}$ = 2672+789+688 = 4149 грн.

$З_м$ = (93+85+4149)/50 = 86,5 грн.

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$О_п$ = 944·15·0.01 = 141 грн.

Визначення витрат на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_в$ = 50 прим.):

$$A_м = \frac{A_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $A_р$ – загальна річна сума амортизаційних відрахувань, грн.

$А_м = 214344 \cdot 1/(50 \cdot 12) = 357$ грн.

Повна собівартість Пз визначається як сума витрат за попередніми статтями калькуляції:

$$C_п = З_о + З_д + C_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$С_п = 944+94,4+378+141,6+86,5+141+357 = 2148,5$грн.

Визначення планового прибутку за рівнем рентабельності ($Р_п$) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 30%.

$$П_р = 0,01 \cdot Р_п \cdot С_п, \quad (7.22)$$

де: рівень $Р_п$ = рівень рентабельності, %.

$П_р = 0,01 \cdot 30 \cdot 2148,5 = 644,5$ грн.

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн.
1	2	3
1. Основна зарплата виконавців	$З_о$	944
2. Додаткова зарплата виконавців	$З_д$	94,4
3. Відрахування на соціальні потреби	$С_{оц}$	384
4. Загальногосподарські витрати	$Г_{осп}$	141,6
5. Витрати на матеріали	$З_М$	26
6. Освоєння нових операційних систем, мов програмування	$О_п$	141
7. Амортизація основних фондів	$А_м$	357

Продовження таблиці 7.9

Найменування статей витрат	*Позначення*	Величина, грн.
1	*2*	3
8. Повна собівартість програмного забезпечення	$С_п$	2148,5
9. Плановий прибуток	$П_р$	644,5
10. Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	2793
11. Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	*ПДВ*	558,6
12. Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	*Ц*	3351,6

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

Об’єм капітальних вкладень у споживача програмної продукції визначаються на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводяться у таблицю 7.9.

Таблиця 7.10 – Розрахунок об’єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	3309
Всього капітальних витрат	–	3309

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаються при умові роботи підсистеми на протязі року. Результати зводяться до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування	$З_р$	16456	11754
2. Витрати на електроенергію	$З_{ел}$	245	175
3. Витрати на амортизацію	$З_{ам}$	0	1655
Всього витрат за рік	$І$	16701	13584

Витрати на обслуговування системи:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_д) \cdot (1 + 0{,}01 \cdot Н_с), \qquad (7.23)$$

де: $Т_р$ — кількість годин обслуговування за рік, год;

$З_г$ — заробітна плата обслуговуючого персоналу, грн/год.

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 140 годин на рік до 100 годин на рік, тому витрати на технічне обслуговування зменшилася з:

$З_{р\,баз} = 140 \cdot 26 \cdot 1{,}1 \cdot 1{,}37 \cdot 3 = 16456$ грн

до:

$З_{р\,нов} = 100 \cdot 26 \cdot 1{,}1 \cdot 1{,}37 \cdot 3 = 11754$ грн

Витрати на електроенергію визначаються з урахуванням споживаємої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \qquad (7.24)$$

$З_{ел\ баз} = 7 \cdot 0,15 \cdot 140 \cdot 1,67 = 245$ грн.

$З_{ел\ нов} = 7 \cdot 0,15 \cdot 100 \cdot 1,67 = 175$ грн.

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку використовується таблиця 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн., за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	50	–	3309	–	1654,5
Всього відрахувань	-	–	3309	–	1654,5

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховується за формулою:

$$E_в = (Ц_п - C_п) \cdot N_е - E_н \cdot K_р , \qquad (7.25)$$

де $К_р$ — балансова вартість основних фондів розробника, грн.

$Е_в = (2793 — 2148,5) \cdot 50\text{-}0,15 \cdot 1217172 \cdot 1/12 = 17010,35$ грн.

Визначення періоду окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_р}{(Ц_п - C_п \cdot N_е)}, \qquad (7.26)$$

$$T_в = \frac{1217172}{(2793 — 2148,5) \cdot 50 \cdot 12 / 1} = 3,14 \text{ років} .$$

Визначення величини економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \quad (7.27)$$

де: $I_б$, $I_н$ — величина експлуатаційних витрат за базовим та новим варіантом відповідно;

$K_б$,$K_н$ — об'єм капітальних вкладень за варіантами, що порівнюються.

$E_{сп}$ = (16701-13584)-0,5·3309 = 1462,5 грн.

Визначення періоду окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \quad (7.28)$$

$$T_{сп} = \frac{3309}{16701 - 13584} = 1{,}06 \ року .$$

Показники економічної ефективності програмної продукції зводяться до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	50
2. Повна собівартість розробленої програми	Грн.	2148,5
3. Ціна розробленої програми	Грн.	2793
4. Плановий прибуток від реалізації розробленої програми	Грн.	644,5
5. Рентабельність програмної продукції	%	30
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1217172
7. Загальний прибуток від реалізації програмної продукції	Грн.	32225

Продовження таблиці 7.13

Найменування показників	Одиниця виміру	Величина
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	17010
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	3,14
10. Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	3309
11. Величина економічного ефекту у користувача програмної продукції	Грн.	1462
12. Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	1,06

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

З метою запобігання пошкоджень, які можуть трапитися через ураження електричним струмом, коротким замиканням, загоряння тощо, було розроблено стандарт безпеки IEC 950. Для країн Європейскої співдружності загальним стандартом електробезпечності є Cemark.

Використання нульового робочого провідника як нульового захисного провідника забороняється. Нульовий захисний провід прокладається від стійки групового розподільчого щита, розподільчого пункту до розеток живлення. Не допускається підключення на щиті до одного контактного затискача нульового робочого та нульового захисного провідників. Площа перерізу нульового робочого та нульового захисного провідника в груповій трипровідній мережі повинна бути не менше площі перерізу фазового провідника.

Програмісти у процесі роботи отримують негативний вплив на органи зору, а також мають значну розумову напругу і нервово-емоційне навантаження. Руки (м’язи рук та суглоби пальців) при роботі з клавіатурою мають теж істотне навантаження. До шкідливих факторів, які впливають на робітників галузі інформаційних технологій спеціалісти відносять високочастотні електромагнітні коливання роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

Розглянемо шкідливі чинники роботи програмістів керуючись наступними нормативно-правовими актами: “Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин” ДсанПіН 3.3.2-007-98, та “Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями” НПАОП 0.00-7.15-18.

Умови праці програміста включають наступні фактори:

- вентиляція приміщення;

- освітлення приміщення;

- параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення негативного впливу комп'ютера на організм людини визначимо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста.[59]

Шкідливі і небезпечні фактори при роботі з комп'ютером

Шкідливими факторами при роботі з персональним комп'ютером є неонізуюче випромінювання промислової частоти, збільшене нервово-емоційне навантаження на оператора, збільшення навантаження на органи зору та дрібні стереостатичні рухи кінцівок.

Ці фактори можуть викликати у працівника певні розлади здоров'я, зокрема підвищення артеріального тиску, кон'юктивіти, тендовагініти ті інші захворювання.

Комп'ютер, як і будь-який електричний прилад, особливо при його неправильному підключенні, може бути джерелом ураження оператора електричним струмом. Саме тому всі працівники, які працюють з персональним комп'ютером, повинні мати першу(або другу) групу допуску з електробезпеки.

Через наявність зазначених факторів працівники, які працюють з персональними комп'ютерами, підлягають попередньому та періодичному медичному огляду згідно з пунктом 6.2.3 додатку 4 до наказу Міністерства охорони здоров'я України "Про затвердження Порядку проведення медичних оглядів працівників певних категорій" від 21 травня 2007 року №246.[61]

8.2 Аналіз умов праці на робочому місці фахівця

Дослідження та аналіз санітарно-гігієнічних умов праці на робочому місці програміста. Чисельність персоналу — 6 осіб.

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1:

Таблиця 8.1 — Розміри приміщення

Найменування	Значення, м
Ширина	6,63
Довжина	8,12
Висота	3,4

Таблиця 8.2 - Площа та обсяг приміщення, на одного працюючого

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	8,9
Об'єм, V	$м^3$	не менше 20.0	30,5

Нормованим параметром природного освітлення являється коефіцієнт природного освітлення (КПО). КПО встановлюється в залежності від розряду виконуваних зорових робіт.

Робота оператора ПК відноситься до робіт середньої точності (IV розряд зорових робіт, мінімальний розмір об'єкту розрізнення складає 0,5 – 1,0 мм), для яких при використанні бокового освітлення КПО=1,5 %.

Для штучного освітлення нормованим параметром виступає Емін – мінімальний рівень освітленості, та Кп – коефіцієнт пульсації світлового потоку, який не повинний бути більшим ніж 20%.

Мінімальна освітленість встановлюється в залежності від розряду виконуваних зорових робіт. Для IV розряду зорових робіт вона складає 300...500 лк.

Була проведена перевірка освітленості робочого місця користувача ПК на відповідність розряду зорової роботи. За даними вимірювань рівень природної

освітленості поверхні, де розташований ПК, складає 200 лк за освітленості тієї же поверхні відкритим небосхилом в 20000 лк, тобто КПО = 1%, що не відповідає нормативному КПО.

Для штучного освітлення у приміщенні використовуються люмінесцентні лампи.[55]

Розташовані у приміщенні 4 ПК є джерелами тепловиділень, крім того для підтримання у приміщенні в холодний період року оптимальних параметрів мікроклімату використовуються нагріті поверхні опалювальної системи. Нормованим показником ІЧВ являється гранично допустима густина потоку енергії $I_{г.д}$, Вт/м^2, яка встановлюється в залежності від площі опромінюваної поверхні тіла людини (Sопр). Нормовані рівні складають:

$I_{г.д}$ =35 Вт/м^2 за $S_{опр}$ > 50%;

$I_{г.д}$ =70 Вт/м^2 за $S_{опр}$ ~ 25-50%;

$I_{г.д}$ =100 Вт/м^2 за $S_{опр}$ < 25%

Нормування параметрів проводиться в залежності від періоду року та категорії важкості виконуваних робіт. Для постійних робочих місць, якими є робочі місця операторів ПК, встановлені оптимальні параметри мікроклімату, а за неможливості їх дотримання використовують допустимі параметри. Робота оператора ПК за енерговитратами відноситься до категорії легких робіт Іа, Іб. В таблиці 8.3[55]. наведені оптимальні параметри мікроклімату в приміщеннях, де виконуються роботи операторського типу.

Таблиця 8.3 - Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Вологість,%	Швидкість повітря, м/с	Температура,° С	Вологість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-23	40-55	0,1
Тепла	23-25	50-70	0,1	24-25	50-65	0,11

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини та попереджають травматизм і професійні захворювання програмістів, є освітлення на робочому місці.

Розробка заходів з умов поліпшення охорони праці

Провівши аналіз умов праці в розглянутому вище приміщенні, було отримано наступні результати:

- значення мікроклімату в приміщенні не перевищує норму;

- розрахунки розміру робочого місця на одного працівника відповідають нормі;

- рівень шуму в приміщенні не становить вище норми.

З вище перечислених результатів можна зробити висновок, що основний вплив на продуктивність ІТ-спеціалістів є його психологічний стан. Тому є доцільним зменшити рівень стресу на робочому місці.

Рекомендовані наступні заходи: За потреби особливої концентрації уваги під час виконання робіт суміжні робочі місця операторів необхідно відділяти одне від одного перегородками висотою 1,5 – 2м. Конструкція робочого місця користувача персонального комп'ютера має забезпечити підтримання оптимальної робочої пози офісного працівника. Конструкція робочого столу має відповідати сучасним вимогам ергономіки і забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання (дисплея, клавіатури, принтера) і документів. Висота робочої поверхні робочого столу має регулюватися в межах 680-800 мм, а ширина і глибина – забезпечувати можливість виконання операцій у зоні

досяжності моторного поля (рекомендовані розміри: 600-1400мм, глибина – 800-1000мм). Робочий стіл повинен мати простір для ніг заввишки не менше ніж 600мм, завширшки не менше ніж 500мм, завглибшки (на рівні колін) не менше ніж 450мм, на рівні простягнутої ноги не менше ніж 650мм. Робочий стілець має бути підйомно-поворотним, регульованим за висотою, з кутом і нахилу сидіння та спинки і за відстанню від спинки до переднього краю сидіння поверхня сидіння має бути плоскою, передній край – заокругленим.[60]

8.3 Розрахункова частина

Занулення — це об’єднання частин електроустановок, які розташовуються не під напругою, із заземленим проводом джерела живлення або з заземленою нейтраллю генератора або обмотки трифазного трансформатора. Тобто — це з’єденання металевої частини електроприладу з нейтраллю трансформатора.

Ефективність роботи занулення визначається чітким та швидким відключенням пошкодженої ділянки електричної мережі при однофазному замиканні на корпус електрообладнання. Швидкість спрацювання захисних пристроїв є залежною від відношення струму короткого струму замикання в місці пошкодження до номінального струму установки відключаючого пристрою.

Розрахунок занулення

Початкові дані

1. Потужність електродвигуна, який підлягає зануленню : P = 10 кВт.

2. Кількість електродвигунів: m = 4.

– Потужність освітлювальних приладів : Po = 30 кВт.

3. Довжина магістрального кабеля: LM=50 м.

4. Довжина розгалудження (від розподільчого щита до електродвигуна) : l=15 м.

5. Материал провідників кабеля−алюміній.

6. Лінійна напруга U=380 В.

7. Фазна напруга Uф=220 В

1.1 Сила номінального струму електроустановки:

$$I_{ном} = \frac{P}{\sqrt{3} \cdot U_{л} \cdot \cos(\varphi)} \quad (8.1)$$

$$I_{ном} = \frac{10 \cdot 1000}{\sqrt{3} \cdot 380 \cdot 0{,}85} = 17.8$$

1.2 Сила пускового струму:

$$I_{пус} = 5 \cdot I \quad (8.2)$$

$$I_{пус} = 5 \cdot 17{,}8 = 89\ A$$

1.3 Номінальна сила струму апарата захисту:

$$I_{н} = \frac{I_{пус}}{\beta} \quad (8.3)$$

$$I_{н} = \frac{89}{2{,}5} = 35{,}6\ A$$

З таблиці 1 [54] вибирається запобіжник ПН2-100 з плавкою вставкою $I_{ном}$ = 40 А.

1.4 Найменше допустиме по умовам спрацьовування захисту значення сили струму короткого замикання, А:

$$I_{kmin} = I_{н} \cdot K \quad (8.4)$$

$$I_{kmin} = 40 \cdot 3 = 120\ A$$

1.5 Переріз провода або кабеля розглалуження з умови допустимого нагрівання:

$$I_{доп} \geq I_{max} \quad (8.5)$$

$$I_{доп} \geq 17,8$$

Площа перерізу узгоджується з номінальним струмом плавкої вставки запобіжника із умови:

$$I_{доп} \geq \frac{I_{вст}}{\alpha} \quad (8.6)$$

$$I_{доп} \geq \frac{40}{3} = 13,3\ A$$

2 Параметри магістрального кабелю:

$$I_{роб} = Ko\sum_{1}^{n}(Kз \quad I_{ном}) \quad (8.7)$$

$$I_{роб} = 0,75 \cdot \left(0,85 \cdot \left(\frac{10 \cdot 1000}{\sqrt{3} \cdot 380 \cdot 0.85}\right) \quad 4 + \frac{10 \cdot 1000}{\sqrt{3} \cdot 380 \cdot 0.85}\right) = 58,74A$$

2.1 Струм короткочасного перевантаження магістрального кабеля:

$$I_{ПЕР} = Ko\sum_{1}^{n-1}(Кз \cdot I_{ном}) + I_{ПУС} \quad (8.8)$$

58,74 + 89 = 147,74 А

2.2 Струм спрацювання теплового або електромагнітного розчеплювача автоматичного вимикача:

$$I_{спр} \geq I_{роб} \geq 58,74 \text{ А} \quad (8.9)$$

2.3 Максимальний струм перевантаження лінії:

$$1,25 \cdot 58,74 = 73,42 \ A \quad (8.10)$$

$I_{спр}$ = 80 А

Вимикач: А3714Б

2.4 Площа перерізу магістрального кабеля:

$$I_{доп} = \frac{I_{СП.Р}}{\alpha} \quad (8.11)$$

$$I_{доп} = \frac{80}{3} = 26,6 \ A$$

3. Потужність трансформатора:

$$N_{ТР} = Ko \frac{\sum_{Д}(K_з \cdot P_{Д.НОМ})}{} \cdot cos\varphi_Д = \frac{Kn\sum P_{Д.НОМ}}{cos\varphi_Д} \quad (8.12)$$

$$N_{ТР} = \frac{0,7 \cdot 180}{0,8} = 157,5\ кВ \cdot А$$

4. Активний та індуктивний опір фазного і нульового захисного провідників на дільницях 1 та 2:

$$R_{ф} = \rho \cdot \left(\frac{L_M}{S_{ф1}}\right) + \rho \cdot \left(\frac{L}{S_{ф2}}\right) \quad (8.13)$$

$$R_{ф} = 0,028 \cdot \left(\frac{85}{95}\right) + 0,028 \cdot \left(\frac{20}{4}\right) = 0,165\ Ом$$

$$R_{п} = \rho \cdot \left(\frac{L_M}{S_{п1}}\right) + \rho \cdot \left(\frac{L}{S_{п2}}\right) \quad (8.14)$$

$$R_{п} = 0,028 \cdot \left(\frac{85}{95}\right) + 0,028 \cdot \left(\frac{20}{3}\right) = 0,234\ Ом$$

5. Дійсне значення струма однофазного короткого замикання

$$I_{кр} = \frac{U_{ф}}{\frac{Z_r}{3} + \sqrt{(R_{ф} + R_{Н.З}) + (X_{ф} + Л_{З.Н} + Л_{Н})^2}} \quad (8.15)$$

$$I_{кр} = \frac{220}{\frac{0,141}{3} + \sqrt{(0,165 + 0,234)^2}} = 493,27\ А$$

6. Максимальна напруга на корпусі обладнання відносно землі при замиканні фази на корпус:

$$U_{кmax} = I_{к} \square\ Z_{н} < U_{доп.д.} \quad (8.16)$$

$$U_{кmax} = 493{,}27 \cdot 0{,}234 = 115{,}42 \; B > 36 \; B$$

Умова не виконується, необхідно замінити запобіжник з плавкою вставкою на автоматичний вимикач із струмовим реле, що дає можливість зменшити час замикання на корпус і підвищити допустиму напругу на корпусі або застосувати повторне заземлення нульового захисного провідника.

Необхідний опір нульового захисного провідника:

$$Rn=(Uдоп \square Ro)/((Ik \square ZH)-Uдоп) \qquad (8.17)$$

$$R_n = 36 \cdot 3/((493{,}27 \cdot 0{,}165) \cdot 36) = 2{,}37 Ом$$

8.4 Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а й також підвищує ефективність виробництва в цілому.

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок захисного штучного заземлення. Розроблено заходи з охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським рівнем) вищої освіти, призначено для системи виявлення вторгнень з використанням алгоритму машинного навчання.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження системи виявлення вторгнень в IoT-інфраструктурі з використанням алгоритму машинного навчання.

Рішення даного завдання полягало у вирішенні наступних задач:

- Був проведений огляд існуючих систем виявлення вторгнень з використанням алгоритму машинного навчання.

- Досліджена система виявлення вторгнень з використанням алгоритму машинного навчання.

- На основі отриманих результатів досліджень створена програмна реалізація системи виявлення вторгнень з використанням алгоритму машинного навчання.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти методи використання декількох алгоритмів навчання моделей та їх порівняння між собою для досягнення більшої точності дозволяють успішно вирішувати завдання систем виявлення вторгнень з використанням алгоритму машинного навчання.

Проведено аналіз предметної галузі в ході якого були виявлені фактори, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудований алгоритм і вибране середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для основного програмного забезпечення, що реалізують його функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосувати протокол SSH.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у промисловій інфраструктурі.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 3309 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 1,06 років.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Як працює система виявлення вторгнень (IDS)? [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://ciksiti.com/uk/chapters/3602-how-does-the-intrusion-detection-system-ids-work---linux.

2. АНАЛІЗ СУЧАСНИХ СИСТЕМ ВИЯВЛЕННЯ ТА ЗАПОБІГАННЯ ВТОРГНЕНЬ В ІНФОРМАЦІЙНО-ТЕЛЕКОМУНІКАЦІЙНИХ СИСТЕМАХ [Електронний ресурс] – Режим доступу до ресурсу: https://ela.kpi.ua/bitstream/123456789/17609/1/meshkov.pdf.

3. IDS – що це таке? Система виявлення вторгнень (IDS) як працює? [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: https://poradumo.com.ua/49510-ids-sho-ce-take-sistema-viiavlennia-vtorgnen-ids-iak-pracuye/.

4. Що таке система виявлення вторгнень (ідентифікаторів)? [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://uk.theastrologypage.com/intrusion-detection-system#menu-3.

5. МЕТОД ВИЯВЛЕННЯ DDOS АТАК НА IOT МЕРЕЖІ [Електронний ресурс] – Режим доступу до ресурсу: http://elar.khmnu.edu.ua/bitstream/123456789/8957/1/НІЧЕПОРУК.pdf.

6. Модель системи виявлення вторгнень для інтелектуальних середовищ [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://er.nau.edu.ua/bitstream/NAU/54497/1/ФККПІ_2021_122_СторощукОА.pdf.

7. ТЕХНОЛОГІЯ РОЗПОДІЛЕНОГО ВИЯВЛЕННЯ ТА ПОПЕРЕДЖЕННЯ ВТОРГНЕНЬ В ІНФОРМАЦІЙНУ СИСТЕМУ ПІДПРИЄМСТВА НА БАЗІ VMWARE NSX [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://dut.edu.ua/repozitorii/ikb/2022/БСДМ-61/КОЛОТУХІН_БСДМ61.pdf.

8. Способи протидії бот-мережам в системах інтернет речей [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: http://ir.nmu.org.ua/bitstream/handle/123456789/154328/Горошко.pdf?sequence=1.

9. Метод виявлення вторгнень в мережу Інтернету речей за допомогою нейромережі [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://ela.kpi.ua/bitstream/123456789/42148/1/Hizzatullin_bakalavr.pdf.

10. WHAT IS CROWDSTRIKE? [Електронний ресурс] – Режим доступу до ресурсу: https://www.crowdstrike.com/products/faq/.

11. CrowdStrike Falcon Review [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://www.fool.com/the-ascent/small-business/endpoint-security/crowdstrike-falcon-review/.

12. What is Snort and how does it work? [Електронний ресурс] – Режим доступу до ресурсу: https://www.techtarget.com/searchnetworking/definition/Snort.

13. Using Snort for intrusion detection [Електронний ресурс]. – 2001. – Режим доступу до ресурсу: https://www.techrepublic.com/article/using-snort-for-intrusion-detection/.

14. What is OSSEC? [Електронний ресурс] – Режим доступу до ресурсу: https://www.blumira.com/glossary/ossec/.

15. Top 10 Best Free Open Source Cyber Security Tools (Pros and Cons) [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://cloudinfrastructureservices.co.uk/top-10-best-free-open-source-cyber-security-tools/.

16. Security Onion: An Interesting Guide For 2021 [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://www.jigsawacademy.com/blogs/cyber-security/security-onion/.

17. Security Onion - Review [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://prezi.com/p/yhu-k53n7myg/security-onion-review/.

18. AIDE (Advanced Intrusion Detection Environment) [Електронний ресурс] – Режим доступу до ресурсу: https://aide.github.io/.

19. Enhancing Linux security with Advanced Intrusion Detection Environment (AIDE) [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://www.redhat.com/sysadmin/linux-security-aide.

20. Samhain Labs overwiev [Електронний ресурс] – Режим доступу до ресурсу: https://www.la-samhna.de/samhain/index.html.

21. Quick overwiev IDS [Електронний ресурс] – Режим доступу до ресурсу: https://indico.cern.ch/event/424682/contributions/1034601/attachments/900145/1269029/elfms_presentation.pdf.

22. Suricata: What is it and how can we use it [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://resources.infosecinstitute.com/topic/suricata-what-is-it-and-how-can-we-use-it/.

23. Evaluation of Intrusion Detection Systems under Denial of Service Attack in virtual Environment [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://www.diva-portal.org/smash/get/diva2:1176622/FULLTEXT02.

24. About Zeek [Електронний ресурс] – Режим доступу до ресурсу: https://docs.zeek.org/en/master/about.html.

25. Intrusion Detection System Techniques and Tools: A Survey [Електронний ресурс] – Режим доступу до ресурсу: https://saspublishers.com/media/articles/SJET53122-130.pdf.

26. What is Python? Executive Summary [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/doc/essays/blurb/.

27. What Is Python Used For? A Beginner's Guide [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python.

28. What is the Python programming language? [Електронний ресурс] – Режим доступу до ресурсу: https://www.techtarget.com/whatis/definition/Python.

29. МЕТОДИ МАШИННОГО НАВЧАННЯ ДЛЯ ВИЯВЛЕННІ АТАК НА КОМП'ЮТЕРНІ СИСТЕМИ [Електронний ресурс] – Режим доступу до ресурсу:

http://elartu.tntu.edu.ua/bitstream/lib/23937/2/V-STC-IMST_2018_Maksymets_O-Machine_learning_methods_101.pdf.

30. 5 Anomaly Detection Algorithms every Data Scientist should know [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://towardsdatascience.com/5-anomaly-detection-algorithms-every-data-scientist-should-know-b36c3605ea16.

31. What is logistic regression? [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/topics/logistic-regression.

32. Що таке випадковий ліс? [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://uk.theastrologypage.com/random-forest.

33. Machine Learning for Anomaly Detection and Categorization in Multi-cloud Environments [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/ftp/arxiv/papers/1812/1812.05443.pdf.

34. Understanding Random Forest [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/#h2_1.

35. Introduction to Random Forest in Machine Learning [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/.

36. k-Nearest Neighbors (kNN) for anomaly detection [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://towardsdatascience.com/k-nearest-neighbors-knn-for-anomaly-detection-fdf8ee160d13.

37. Support Vector Machine (SVM) for Anomaly Detection [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://towardsdatascience.com/support-vector-machine-svm-for-anomaly-detection-73a8d676c331.

38. Fast learning network: A novel artificial neural network with a fast learning speed [Електронний ресурс]. – 2013. – Режим доступу до ресурсу: https://www.researchgate.net/publication/257436050_Fast_learning_network_A_novel_artificial_neural_network_with_a_fast_learning_speed.

39. Extreme learning machine: Theory and applications [Електронний ресурс]. – 2006. – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/abs/pii/S0925231206000385?via%3Dihub .

40. Network intrusion detection system: A systematic study of machine learning and deep learning approaches [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://onlinelibrary.wiley.com/doi/full/10.1002/ett.4150.

41. A New Intrusion Detection System Based on Fast Learning Network and Particle Swarm Optimization [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/8326489.

42. Classifier Performance Evaluation for Lightweight IDS Using Fog Computing in IoT Security [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://www.researchgate.net/publication/353115751_Classifier_Performance_Evaluation_for_Lightweight_IDS_Using_Fog_Computing_in_IoT_Security.

43. Deep Learning for Intrusion Detection and Security of Internet of Things (IoT): Current Analysis, Challenges, and Possible Solutions [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://www.hindawi.com/journals/scn/2022/4016073/.

44. A Machine Learning-Based Lightweight Intrusion Detection System for the Internet of Things [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.researchgate.net/publication/337453353_A_Machine_Learning-Based_Lightweight_Intrusion_Detection_System_for_the_Internet_of_Things.

45. ЩО ТАКЕ SSH? [Електронний ресурс] – Режим доступу до ресурсу: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-ssh/.

46. What is SSH in IoT? [Електронний ресурс] – Режим доступу до ресурсу: https://www.quora.com/What-is-SSH-in-IoT.

47. SSH and Intrusion Detection [Електронний ресурс] – Режим доступу до ресурсу: https://www.giac.org/paper/gsec/1677/ssh-intrusion-detection/103050.

48. Intrusion Detection System using SSH Protocol [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: http://www.ijirset.com/upload/2015/november/130_Intrusion.pdf.

49. SSH and IDS [Електронний ресурс]. – 2002. – Режим доступу до ресурсу: https://www.sans.org/white-papers/358/.

50. Cybersecurity of Internet of Things Devices: A Secure Shell Implementation [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://www.theseus.fi/bitstream/handle/10024/748434/Timko_Alexander.pdf?sequence=2.

51. Система виявлення мережевих втручань з використанням машинного навчання [Електронний ресурс] – Режим доступу до ресурсу: https://ela.kpi.ua/bitstream/123456789/34796/1/Savosko-O-M_bakalavr.pdf.

52. Python® – the language of today and tomorrow [Електронний ресурс] – Режим доступу до ресурсу: https://pythoninstitute.org/about-python.

53. PyCharm Features [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/pycharm/features/.

54. Методичні вказівки по виконанню розрахунків з використанням персональних ЕОМ IBM−сумісного типу, 2−ге видання, перероблене та доповнене [Електронний ресурс] – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/bitstream/123456789/8769/1/Zanul_2019_pub.pdf.(дата звернення: 30.11.2022)

55. Аналіз умов праці на робочому місці користувача ПК [Електронний ресурс] – Режим доступу до ресурсу: https://cpo.stu.cn.ua/Oksana/dipl_bak/140.html.

56. Розрахунок опору природного заземлювача. Розрахунок заземлюючих пристроїв. Послідовне розташування електродів [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://tigerdoor.ru/uk/dizajjn/raschet-soprotivleniya-estestvennogo-zazemlitelya-raschet/.

57. Розрахунки з електробезпеки [Електронний ресурс] – Режим доступу до ресурсу: https://cpo.stu.cn.ua/Oksana/rozrah_rozd_OP_DP_bak_spec_mag/90.html.

58. Сакулин В.П., Шептовицкий В.М. Безопасность труда при монтаже и эксплуатации электроустановок / В.П.Сакулин, В.М.Шептовицкий. – Л. : “Колос”, 1973. – 238 с.

59. Вимоги до електробезпеки у офісних приміщеннях з комп’ютерною технікою [Електронний ресурс] – Режим доступу до ресурсу: https://cpo.stu.cn.ua/Oksana/posibnik/1140.html.

60. Охорона праці в офісі. Вимоги до робочого місця офісного працівника [Електронний ресурс] – Режим доступу до ресурсу: https://gc.ua/uk/oxorona-praci-v-ofisi-vimogi-do-robochogo-miscya-ofisnogo-pracivnika/.

61. Які шкідливі та небезпечні фатокри виникають при роботі з комп’ютером [Електронний ресурс] – Режим доступу до ресурсу: https://oppb.com.ua/news/stvorennya-spryyatlyvyh-umov-praci-na-vyrobnyctvi.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРМ-123.22.0029.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Шовкопляс Ю.С.			Дослідження та програмна реалізація системи виявлення вторгнень в промислову IoT-інфраструктуру	Літ.	Аркуш	Аркушів
Перевірив		Якименко Н.М..				М	1	6
Н. Контр.		Гермак В.С.				ЦНТУ КІ-21М-1,4		
Затв.		Смірнов О.А.						

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи виявлення вторгнень в промислову IoT-інфраструктуру.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 19-13 від 17.08.2022 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи виявлення вторгнень в промислову IoT-інфраструктуру.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи виявлення вторгнень в промислову IoT-інфраструктуру;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Python 3.9.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2022 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинна бути розглянута пожежна безпека.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 98 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2022 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 21.12.2022 р.

Додаток Б

(обов'язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи

за другим (магістерським) рівнем вищої освіти

_____ Якименко Н.М.

Дослідження та програмна реалізація системи виявлення вторгнень в промислову IoT-інфраструктуру

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 50

Літера: РП

Кропивницький – 2022 року

file IDS.py

```
import numpy as np
import pandas as pd
from sklearn.kernel_approximation import RBFSampler
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import classification_report
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from     sklearn.metrics     import     (precision_score,     recall_score,f1_score,
accuracy_score,mean_squared_error,mean_absolute_error)
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import Normalizer
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from     sklearn.metrics     import     (precision_score,     recall_score,f1_score,
accuracy_score,mean_squared_error,mean_absolute_error,                roc_curve,
classification_report,auc)

traindata = pd.read_csv('kddtrain.csv', header=None)
testdata = pd.read_csv('kddtest.csv', header=None)

X = traindata.iloc[:,1:42]
Y = traindata.iloc[:,0]
C = testdata.iloc[:,0]
T = testdata.iloc[:,1:42]

scaler = Normalizer().fit(X)
trainX = scaler.transform(X)

scaler = Normalizer().fit(T)
testT = scaler.transform(T)


traindata = np.array(trainX)
trainlabel = np.array(Y)

testdata = np.array(testT)
testlabel = np.array(C)



#traindata = X_train
#testdata = X_test
#trainlabel = y_train
#testlabel = y_test

print("----------------------------------------LR-----------------------------------
")
model = LogisticRegression()
model.fit(traindata, trainlabel)
```

```
# make predictions
expected = testlabel
np.savetxt('classical/expected.txt', expected, fmt='%01d')
predicted = model.predict(testdata)
proba = model.predict_proba(testdata)

np.savetxt('classical/predictedlabelLR.txt', predicted, fmt='%01d')
np.savetxt('classical/predictedprobaLR.txt', proba)

y_train1 = expected
y_pred = predicted
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")

print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)

# fit a Naive Bayes model to the data
print("-----------------------------------------NB-------------------------------
")
model = GaussianNB()
model.fit(traindata, trainlabel)
print(model)
# make predictions
expected = testlabel
predicted = model.predict(testdata)
proba = model.predict_proba(testdata)

np.savetxt('classical/predictedlabelNB.txt', predicted, fmt='%01d')
np.savetxt('classical/predictedprobaNB.txt', proba)

y_train1 = expected
y_pred = predicted
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")

print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)

# fit a k-nearest neighbor model to the data
print("-----------------------------------------KNN-------------------------------
-")
model = KNeighborsClassifier()
```

```
model.fit(traindata, trainlabel)
print(model)
# make predictions
expected = testlabel
predicted = model.predict(testdata)
proba = model.predict_proba(testdata)


np.savetxt('classical/predictedlabelKNN.txt', predicted, fmt='%01d')
np.savetxt('classical/predictedprobaKNN.txt', proba)

# summarize the fit of the model

y_train1 = expected
y_pred = predicted
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")


print("-----------------------------------------------")
print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)

print("---------------------------------------DT--------------------------------
")

model = DecisionTreeClassifier()
model.fit(traindata, trainlabel)
print(model)
# make predictions
expected = testlabel
predicted = model.predict(testdata)
proba = model.predict_proba(testdata)

np.savetxt('classical/predictedlabelDT.txt', predicted, fmt='%01d')
np.savetxt('classical/predictedprobaDT.txt', proba)
# summarize the fit of the model

y_train1 = expected
y_pred = predicted
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")


print("-----------------------------------------------")
print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
```

```
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)


print("----------------------------------------Adaboost---------------------------
------")

model = AdaBoostClassifier(n_estimators=100)
model.fit(traindata, trainlabel)

# make predictions
expected = testlabel
predicted = model.predict(testdata)
proba = model.predict_proba(testdata)

np.savetxt('classical/predictedlabelAB.txt', predicted, fmt='%01d')
np.savetxt('classical/predictedprobaAB.txt', proba)
# summarize the fit of the model

y_train1 = expected
y_pred = predicted
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")


print("-----------------------------------------------")
print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)

model = RandomForestClassifier(n_estimators=100)
model = model.fit(traindata, trainlabel)

# make predictions
expected = testlabel
predicted = model.predict(testdata)
proba = model.predict_proba(testdata)
np.savetxt('classical/predictedlabelRF.txt', predicted, fmt='%01d')
np.savetxt('classical/predictedprobaRF.txt', proba)

# summarize the fit of the model

print("----------------------------------------RF---------------------------------
--")

y_train1 = expected
y_pred = predicted
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
```

```
f1 = f1_score(y_train1, y_pred, average="binary")


print("-----------------------------------------------")
print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)


model = svm.SVC(kernel='rbf',probability=True)
model = model.fit(traindata, trainlabel)

# make predictions
expected = testlabel
predicted = model.predict(testdata)
proba = model.predict_proba(testdata)
np.savetxt('classical/predictedlabelSVM-rbf.txt', predicted, fmt='%01d')
np.savetxt('classical/predictedprobaSVM-rbf.txt', proba)

print("---------------------------------------SVMrbf--------------------------------
------")
y_train1 = expected
y_pred = predicted
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")

print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)


model = svm.SVC(kernel='linear', C=1000,probability=True)
model.fit(traindata, trainlabel)
print(model)
# make predictions
expected = testlabel
predicted = model.predict(testdata)
proba = model.predict_proba(testdata)

np.savetxt('classical/predictedlabelSVM-linear.txt', predicted, fmt='%01d')
np.savetxt('classical/predictedprobaSVM-linear.txt', proba)

# summarize the fit of the model
print("---------------------------------------SVM linear----------------------------
----------")
y_train1 = expected
y_pred = predicted
```

```
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")

print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)

traindata = pd.read_csv('expected.txt', header=None)

testdata = pd.read_csv('predictedlabelAB.txt', header=None)



y_train1 = traindata
y_pred = testdata
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")

print("ABresults")
print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)


testdata = pd.read_csv('predictedlabelDT.txt', header=None)



y_train1 = traindata
y_pred = testdata
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")

print("DTresults")
print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)
```

```
testdata = pd.read_csv('predictedlabelKNN.txt', header=None)



y_train1 = traindata
y_pred = testdata
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")

print("KNNresults")
print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)



testdata = pd.read_csv('predictedlabelLR.txt', header=None)



y_train1 = traindata
y_pred = testdata
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")

print("LRresults")
print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)



testdata = pd.read_csv('predictedlabelNB.txt', header=None)



y_train1 = traindata
y_pred = testdata
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")

print("NBresults")
```

```
print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)



testdata = pd.read_csv('predictedlabelRF.txt', header=None)



y_train1 = traindata
y_pred = testdata
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")

print("RFresults")
print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)



testdata = pd.read_csv('predictedlabelSVM-linear.txt', header=None)



y_train1 = traindata
y_pred = testdata
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")

print("SVM-linearresults")
print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)



testdata = pd.read_csv('predictedlabelSVM-rbf.txt', header=None)
```

```
y_train1 = traindata
y_pred = testdata
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")

print("SVM-rbfresults")
print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)

testdata = pd.read_csv('dnnres/dnn1predicted.txt', header=None)
traindata = pd.read_csv('dnnres/expected.txt', header=None)


y_train1 = traindata
y_pred = testdata
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")

print("dnn1results")
print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)

print("/n")

testdata = pd.read_csv('dnnres/dnn2predicted.txt', header=None)
traindata = pd.read_csv('dnnres/expected.txt', header=None)


y_train1 = traindata
y_pred = testdata
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")

print("dnn2results")
print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("racall")
```

```
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)
print("/n" )


testdata = pd.read_csv('dnnres/dnn3predicted.txt', header=None)
traindata = pd.read_csv('dnnres/expected.txt', header=None)


y_train1 = traindata
y_pred = testdata
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")

print("dnn3results")
print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)
print("/n" )


testdata = pd.read_csv('dnnres/dnn4predicted.txt', header=None)
traindata = pd.read_csv('dnnres/expected.txt', header=None)


y_train1 = traindata
y_pred = testdata
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")

print("dnn4results")
print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)
print("/n" )

testdata = pd.read_csv('dnnres/dnn5predicted.txt', header=None)
traindata = pd.read_csv('dnnres/expected.txt', header=None)


y_train1 = traindata
y_pred = testdata
accuracy = accuracy_score(y_train1, y_pred)
recall = recall_score(y_train1, y_pred , average="binary")
```

```
precision = precision_score(y_train1, y_pred , average="binary")
f1 = f1_score(y_train1, y_pred, average="binary")

print("dnn5results")
print("accuracy")
print("%.3f" %accuracy)
print("precision")
print("%.3f" %precision)
print("racall")
print("%.3f" %recall)
print("f1score")
print("%.3f" %f1)
print("/n" )
```

file numpy.py

```
files_len = counter = 0
main = None
update_button = None
scan_button = None
fullscan_button = None
quit_button = None
b_delete = None
b_delete_all = None
b_restore = None
b_restore_all = None
b_add_file = None
text_box = None
e = None
li = None
rb1 = None
rb2 = None
method = None
bgc = None
fgc = None
special = None
special_text = None
t_time = None

daytime = int(time.strftime("%H", time.localtime()))




def clock_thread():
    global e

    months = ["January", "February", "March", "April", "May", "June", "Juli",
"August", "September", "October", "November", "December"]
    while True:
        string_time = "%H:%M:%S o'clock, on %d.{0}.%Y"
        month_name = time.strftime("%B", time.localtime())
```

```
        for i in range(len(months)):
            if months[i] == month_name:
                month_name = str(i+1)
                if int(month_name) < 10:
                    month_name = "0" + month_name
                break
        string_time = string_time.format(month_name)
        current_time = time.strftime(string_time, time.localtime())
        e.delete(0, len(e.get()))
        e.update()
        e.insert(0, current_time)
        e.update()
        time.sleep(1)

def ScanSystemFiles():
    global files
    global text_box
    global files_len

    text_box.insert(END, "[ * ] Scanning system for files...\n")
    text_box.see(END)
    text_box.update()
    time.sleep(3)
    text_box.see(END)
    text_box.update()
    SystemFileScanner.partitions(partitionen_folder)
    f = open(partitionen_folder, "r")
    content = f.read()
    f.close()
    content = content.splitlines()
    files = content
    files_len = len(files)
    text_box.insert(END, "[ + ] System successfully prepared\n", 'positive')
    text_box.tag_config("positive", foreground="green")
    text_box.see(END)
    text_box.update()

def full_scan(part):
    global verzeichnisse
    global files
    global text_box
    global e
    global full_scan
    global files_len
    global lock
    global t_time
    global counter

    if part == 1:#Thread-1
        i = int(len(files)*0.125)
```

```
        tmp = 0
    if part == 2:#Thread-2
        i = int(len(files)*0.25)
        tmp = int(len(files)*0.125)
    if part == 3:#Thread-3
        i = int(len(files)*0.375)
        tmp = int(len(files)*0.25)
    if part == 4:#Thread-4
        i = int(len(files)*0.5)
        tmp = int(len(files)*0.375)
    if part == 5:#Thread-5
        i = int(len(files)*0.625)
        tmp = int(len(files)*0.5)
    if part == 6:#Thread-6
        i = int(len(files)*0.75)
        tmp = int(len(files)*0.625)
    if part == 7:#Thread-7
        i = int(len(files)*0.875)
        tmp = int(len(files)*0.75)
    if part == 8:#Thread-8
        i = int(len(files))
        tmp = int(len(files)*0.875)

    if len(files) == 0:
        return ScanSystemFiles()

    text_box.tag_config('positive', foreground="green")
    text_box.see(END)
    text_box.update()
    counter = 0
    st = 0
    while i >= tmp:
        try:
            f = open(files[i], "rb")
            file_content = f.read()
            f.close()
        except:
            continue
        ret = scan_auto(files[i])
        if ret == True:
            text_box.insert(END, "[ ! ] Program: " + files[i] + " might be
dangerous\n", "important")
            text_box.tag_config("important", foreground="red")
            text_box.see(END)
            text_box.update()
            quarantaene.encode_base64(files[i])
        files_len -= 1
        i -= 1
    runtime = int(time.time() - start)
```

```
    text_box.insert(END, "[ + ] Scan ended after\n " + str(runtime/60) + "
minutes.\n", "positive")
    text_box.tag_config("positive", foreground="green")
    if files_len == 0:
        full_scan["state"] = "normal"
    if len(terminations) == 0:
        text_box.insert(END, "[ +++ ] Your PC is safe" + "\n", 'important')
    else:
        text_box.insert(END, "[ !!! ] Found {0} Threats on your
PC\n".format(len(terminations)))
    text_box.tag_config("important", background="red")
    text_box.see(END)
    text_box.update()

def quarantine():
    global text_box
    global terminations
    global li
    global b_delete
    global b_delete_all
    global b_restore
    global b_restore_all
    global b_add_file


    k = 0
    while True:
        tmp = len(li.get(k))
        if tmp == 0:
            break
        else:
            li.delete(0, tmp)
            k += 1
    li.update()



    terminations = glob.glob(quarantine_folder)
    if terminations == []:
        text_box.insert(END, "[ + ] No files in quarantine\n", "positive")
        text_box.tag_config('positive', foreground="green")
        text_box.see(END)
        text_box.update()
    else:
        text_box.insert(END, "[ + ] Files in quarantine:\n", "positive")
        text_box.tag_config('positive', foreground="green")
        text_box.see(END)
        text_box.update()
        for i in terminations:
            text_box.insert(END, "[ * ] " + i + "\n", "info")
            text_box.tag_config("info", background = "red")
```

```
            text_box.see(END)
            text_box.update()
            li.insert(END, i)
            li.update()

    b_delete_all["command"] =lambda:button_action_handler("delete_all")
    b_delete["command"] = lambda:button_action_handler("delete")
    b_restore["command"] = lambda:button_action_handler("restore")
    b_restore_all["command"] = lambda:button_action_handler("restore_all")
    b_add_file["command"] = lambda:button_action_handler("add_file")


def delete(file, ALL):#ALL = 1 => deletes all objects in quarantine
    global li
    global text_box
    global terminations

    if len(terminations) != 0:
        if ALL == 1:
            for i in range(len(terminations)):
                os.remove(terminations[i])
                text_box.insert(END,   "[   +   ]   Deletion   successful:   \n"   +
terminations[i] + "\n", "positive")
                text_box.tag_config("positive", foreground="green")
                text_box.see(END)
                text_box.update()
                li.delete(0, len(terminations[i]))
                li.update()
        elif ALL == 0:
            os.remove(file)
            li.delete(ACTIVE, len(file))
            li.update()
            text_box.insert(END, "[ + ] Deletion successful:\n" + file + "\n",
"positive")
            text_box.tag_config("positive", foreground="green")
            text_box.see(END)
            text_box.update()

        terminations = glob.glob(quarantine_folder)
        for i in terminations:
            li.insert(END, i)
        li.update()
    else:
        text_box.insert(END, "[ - ] Unable to locate any files\n", "negative")
        text_box.tag_config("negative", foreground="red")
        text_box.see(END)
        text_box.update()

def restore(file, ALL):
    global li
```

```
    global text_box
    global terminations

    if len(terminations) != 0:
        if ALL == 1:
            for i in range(len(terminations)):
                quarantaene.decode_base64(terminations[i])
                text_box.insert(END,   "[   +   ]   Successfully   restored\n"   +
terminations[i] + "\n", 'positive')
                text_box.tag_config('positive', foreground="green")
                text_box.see(END)
                text_box.update()
                li.delete(0, len(terminations[i]))
                li.update()
        elif ALL == 0:
            quarantaene.decode_base64(file)
            li.delete(ACTIVE, len(file))
            text_box.insert(END, "[ + ] Successfully restored\n" + file + "\n",
"positive")
            text_box.tag_config("positive", foreground="green")
            text_box.see(END)
            text_box.update()

        terminations = glob.glob(quarantine_folder)
        for i in terminations:
            li.insert(END, i)
        li.update()

    else:
        text_box.insert(END, "[ - ] Unable to locate any files\n", "negative")
        text_box.tag_config("negative", foreground="red")
        text_box.see(END)
        text_box.update()


def add_file_to_quarantine():
    global li
    global terminations

    file = askopenfilename()
    file = file.replace("/", "\\")
    quarantaene.encode_base64(file, file_to_quarantine)
    text_box.insert(END, "[ + ] Moved to quarantine:\n" + file + "\n", "positive")
    text_box.tag_config("positive", foreground="green")
    text_box.see(END)
    text_box.update()
    li.update()

    k = 0
    while True:
```

```
            tmp = len(li.get(k))
            if tmp == 0:
                break
            else:
                li.delete(0, tmp)
                k += 1
        li.update()

        terminations = glob.glob(quarantine_folder)
        for i in terminations:
            li.insert(END, i)
            li.update()

def scan_auto(file):
        time.sleep(3)
        try:
            f = open(file, "rb")
            content = f.read()
            f.close()
            content = create_md5(content)
        except MemoryError:
            f.close()
            return False
        except:
            f.close()
            return False

        signatures = open(large_signatures, "rb")
        try:
            if content in signatures.read():#fastest solution
                signatures.close()
                return True
            else:
                signatures.close()
                return False
        except MemoryError:
            try:
                signatures.close()
                signatures = open(large_signatures, "rb")
                if content in signatures.readlines():#again fast, but around 4 times
slower than the fastest
                    signatures.close()
                    return True
                else:
                    signatures.close()
                    return False
            except MemoryError:
                signatures.close()
                signatures = open(large_signatures, "rb")
                while True:#slowest solution, but can read files sized over 2 GB
                    tmp = signatures.readline()
```

```
                if tmp == b"":
                    signatures.close()
                    break

                if tmp == content:
                    signatures.close()
                    return True
            return False
    except:
        return False

def scan():
    global text_box

    match = False
    file = askopenfilename()
    start = time.time()
    text_box.insert(END, "[ * ] Scanning " + file + "\n")
    text_box.see(END)
    text_box.update()
    try:
        f = open(file, "rb")
        content = f.read()
        f.close()
        content = create_md5(content)
        text_box.insert(END, "MD5-Hash: " + content.decode("utf-8") + "\n")
        text_box.see(END)
        text_box.update()
    except MemoryError:
        text_box.insert(END,   "[   -   ]   Unable   to   create   MD5-Hash:\n-----
>MemoryError!\n", 'negative')
        text_box.insert(END, "[ ! ] Only select files under 1 GB\n", "negative")
        text_box.tag_config('negative', foreground="red")
        text_box.see(END)
        text_box.update()
        return None
    except Exception as e:
        text_box.insert(END, "[ ! ] Unable to handle problem\n[ ! ] Try again/file
might be corrupted\n", "negative")
        text_box.tag_config('negative', foreground="red")
        text_box.see(END)
        text_box.update()
        return None

    signatures = open(large_signatures, "rb")

    try:
        if content in signatures.read():#fastest solution
            signatures.close()
            match = True
```

```
        else:
            match = False
            signatures.close()
    except MemoryError:
        try:
            signatures.close()
            signatures = open(large_signatures, "rb")
            if content in signatures.readlines():#again fast, but around 4 times
slower than the fastest
                f.close()
                match = True
            else:
                signatures.close()
                match = False
        except MemoryError:
            signatures.close()
            signatures = open(large_signatures, "rb")
            while True:#slowest solution, but can read files sized over 2 GB
                tmp = signatures.readline()
                if tmp == b"":
                    signatures.close()
                    break
                if tmp == content:
                    match = True
                    signatures.close()
    except:
        text_box.insert(END, "[ - ] Something bad happened while performing the
task\n", "negative")
        text_box.tag_config("negative", foreground="red")
        text_box.see(END)
        text_box.update()
        return None


    text_box.insert(END, "[ * ] Scan duration: {0}\n".format(round(time.time()-
start, 2)))
    text_box.see(END)
    text_box.update()
    if match:
        quarantaene.encode_base64(file, file_to_quarantine)
        text_box.insert(END, "[ ! ] Threat found: {0}\n[ ! ] File was moved into
quarantine","important")
        text_box.tag_config("important", foreground="red")
        text_box.see(END)
        text_box.update()
    if not match:
        text_box.insert(END, "[ + ] No threat was found\n", "positive")
        text_box.tag_config("positive", foreground="green")
        text_box.see(END)
        text_box.update()
```

```
def closing():
    main.destroy()
    sys.exit()

def button_action_handler(s):
    global files_len
    global text_box
    global t_time
    global fullscan_button
    global b_delete
    global b_delete_all
    global b_restore
    global b_restore_all
    global b_add_file
    global li
    global rb1
    global rb2
    global method

    if s == "rb1":
        method = 1
        rb1.place_forget()
        rb2.place_forget()
    if s == "rb2":
        method = 2
        rb2.place_forget()
        rb1.place_forget()

    if s == "delete":
        tb = Thread(target=delete, args=(li.get(ACTIVE),0))
        tb.start()
    if s == "delete_all":
        tb = Thread(target=delete, args=(0,1))
        tb.start()
    if s == "restore":
        tb = Thread(target=restore, args=(li.get(ACTIVE),0))
        tb.start()
    if s == "restore_all":
        tb = Thread(target=restore, args=(0,1))
        tb.start()

    if s == "add_file":
        tb = Thread(target=add_file_to_quarantine)
        tb.start()

    if s == "update_button":
        tb = Thread(target=link_collector)
        tb.start()
```

```
    if s == "scan_button":
        tb = Thread(target=scan)
        tb.start()

    if s == "fullscan_button":
        if files_len == 0:
            text_box.insert(END, "[ ! ] Preparing program\n", "important")
            text_box.see(END)
            text_box.update()
        elif files_len < len(files):
            text_box.insert(END, "[ ! ] One scan is already in action\n",
"important")
            text_box.see(END)
            text_box.update()
        else:
            fullscan_button["state"] = "disabled"
            t_time = time.time()
            text_box.insert(END, "[ ! ] Got {0} files to scan\n".format(files_len),
'important')
            text_box.tag_config("important", foreground="red")
            text_box.update()
            text_box.insert(END, "[ * ] Scan might last for hours...\n")
            text_box.see(END)
            text_box.update()
            tb1 = Thread(target=full_scan, args=(1,))
            tb1.start()
            time.sleep(1)
            tb2 = Thread(target=full_scan, args=(2,))
            tb2.start()
            time.sleep(1)
            tb3 = Thread(target=full_scan, args=(3,))
            tb3.start()
            time.sleep(1)
            tb4 = Thread(target=full_scan, args=(4,))
            tb4.start()
            time.sleep(1)
            tb5 = Thread(target=full_scan, args=(5,))
            tb5.start()
            time.sleep(1)
            tb6 = Thread(target=full_scan, args=(6,))
            tb6.start()
            time.sleep(1)
            tb7 = Thread(target=full_scan, args=(7,))
            tb7.start()
            time.sleep(1)
            tb8 = Thread(target=full_scan, args=(8,))
            tb8.start()

    if s == "quarantine_button":
        if li.winfo_viewable()  == 0:
            b_delete.place(x = 570, y = 70)
```

```
            b_delete_all.place(x = 570, y = 95)
            b_restore.place(x = 570, y = 120)
            b_restore_all.place(x = 570, y = 145)
            b_add_file.place(x = 570, y = 170)
            li.place(x = 570, y = 18.5)
            tb = Thread(target=quarantine)
            tb.start()
        if li.winfo_viewable() == 1:
            b_delete.place_forget()
            b_delete_all.place_forget()
            b_restore.place_forget()
            b_restore_all.place_forget()
            b_add_file.place_forget()
            li.place_forget()

    if s == "quit_button":
        tb = Thread(target=closing)
        tb.start()

def gui_thread():
    global main
    global update_button
    global scan_button
    global fullscan_button
    global quit_button
    global text_box
    global e
    global files_len
    global files
    global li
    global b_delete
    global b_delete_all
    global b_restore
    global b_restore_all
    global b_add_file
    global rb1
    global rb2
    global method
    global bgc
    global fgc
    global special_text

    main = tkinter.Tk()
    main.title("AntiVirus")
    main.wm_iconbitmap("")
    main.configure(bg=bgc)
    main.geometry("750x205")#width x height
    main.resizable(False, False)

    hoehe = 2
```

```
    breite = 20



    update_button = tkinter.Button(main, bg=bgc, fg=fgc, text = "Update",
command=lambda:button_action_handler("update_button"), height = hoehe, width =
breite)
    update_button.grid(row = 0, column = 0)
    scan_button = tkinter.Button(main, bg=bgc, fg=fgc, text = "Scan",
command=lambda:button_action_handler("scan_button"), height = hoehe, width =
breite)
    scan_button.grid(row = 1, column = 0)
    fullscan_button = tkinter.Button(main, bg=bgc, fg=fgc, text = "Full scan",
command=lambda:button_action_handler("fullscan_button"), height = hoehe, width =
breite)
    fullscan_button.grid(row = 2, column = 0)
    quarantine_button = tkinter.Button(main, bg=bgc, fg=fgc, text = "Quarantine",
command=lambda:button_action_handler("quarantine_button"), height = hoehe, width =
breite)
    quarantine_button.grid(row = 3, column = 0)
    quit_button = tkinter.Button(main, bg=bgc, fg=fgc, text = "Close",
command=lambda:button_action_handler("quit_button"), height = hoehe, width =
breite)
    quit_button.grid(row = 4, column = 0, sticky="w")
    b_delete = tkinter.Button(main, bg=bgc, fg=fgc, text = "Remove current",
height=0, width = 25, justify=CENTER)
    b_delete_all = tkinter.Button(main, bg=bgc, fg=fgc, text = "Remove all", height
= 0, width = 25, justify=CENTER)
    b_restore = tkinter.Button(main, bg=bgc, fg=fgc, text = "Restore current",
height=0, width = 25, justify=CENTER)
    b_restore_all = tkinter.Button(main, bg=bgc, fg=fgc, text = "Restore all",
height = 0, width = 25, justify=CENTER)
    b_add_file = tkinter.Button(main, bg=bgc, fg=fgc, text = "Add file", height =
0, width = 25, justify=CENTER)
    b_delete.place(x = 570, y = 70)
    b_delete_all.place(x = 570, y = 95)
    b_restore.place(x = 570, y = 120)
    b_restore_all.place(x = 570, y = 145)
    b_add_file.place(x = 570, y = 170)
    b_delete.place_forget()
    b_delete_all.place_forget()
    b_restore.place_forget()
    b_restore_all.place_forget()
    b_add_file.place_forget()

    #Text
    text_box = tkinter.scrolledtext.ScrolledText(main)
    text_box.configure(bg=bgc)
    text_box.configure(fg=fgc)
    text_box.place(height = 205, width = 419,x = 150, y = 0)
```

```
    li = tkinter.Listbox(main, height=3, width = 29)
    li.place(x = 570, y = 18.5)
    li.place_forget()


    e = tkinter.Entry(main,width = 30)
    e.place(x = 570, y = 0)
    e["justify"] = CENTER
    e.insert(0, "")
    e["bg"] = bgc
    e["fg"] = fgc


    text_box.insert(END, special_text, "VIP")
    text_box.tag_config("VIP", background=special)
    text_box.insert(END, "[ + ] Preparing the program\n", 'positive')
    text_box.tag_config('positive', foreground='green')
    text_box.see(END)
    text_box.update()
    text_box.insert(END, "[ ! ] You might have to wait for a bit\n", 'important')
    text_box.tag_config('important', foreground="red")
    text_box.see(END)
    text_box.update()
    #row_counter += 3
    main.mainloop()


t_main = Thread(target=gui_thread)# Main Thread
t_files = Thread(target=ScanSystemFiles)
t_clock = Thread(target=clock_thread)
t_main.start()
time.sleep(1)
t_clock.start()
time.sleep(5)
#print(t_main.isAlive())
t_files.start()

os_name = sys.platform
partitionen = []
verzeichnisse = []
files = []

def partitions(sfsFolder):
    global partitionen
    big = 65

    if "win" in os_name:
        for i in range(26):
            try:
                if glob.glob(str(chr(big + i)) + ":\\"):
```

```
                    #print("Successfully found partition: " + str(chr(big + i)))
                    partitionen.append(str(chr(big + i)) + ":\\")
            except:
                continue
        return indeces(sfsFolder)
    if "win" not in os_name:
        return indeces(sfsFolder)

def indeces(sfsFolder):
    global verzeichnisse
    global files

    if "win" in os_name:
        verzeichnisse2 = glob.glob("\\*")
    else:
        verzeichnisse2 = glob.glob("//*")
    verzeichnisse_tmp = []
    x = 1

    if "win" in os_name:
        for ind in range(len(partitionen)):
            #print(partitionen[ind])
            while verzeichnisse2 != []:
                verzeichnisse2 = glob.glob(partitionen[ind] + "\\*"*x)
                for i in range(len(verzeichnisse2)):
                    verzeichnisse.append(verzeichnisse2[i])
                x += 1
            x = 1

        for i in range(len(verzeichnisse)):
            if "." in verzeichnisse[i]:
                files.append(verzeichnisse[i])
        for i in range(len(verzeichnisse)):
            if not os.path.isfile(verzeichnisse[i]):
                verzeichnisse_tmp.append(verzeichnisse[i])
        verzeichnisse = verzeichnisse_tmp
        i = 0
        f = open(sfsFolder, "w")
        for i in range(len(files)):
            f.write(files[i] + "\n")
        f.close()
        time.sleep(3)

    if "win" not in os_name:
        while verzeichnisse2 != []:
            verzeichnisse = glob.glob("//*" * x)
            for i in range(len(verzeichnisse2)):
                verzeichnisse.append(verzeichnisse2[i])
            x += 1
        x = 1
```

```
        for i in range(len(verzeichnisse)):
            if "." in verzeichnisse[i]:
                files.append(verzeichnisse[i])
        for i in range(len(verzeichnisse)):
            if not os.path.isfile(verzeichnisse[i]):
                verzeichnisse_tmp.append(verzeichnisse[i])
        verzeichnisse = verzeichnisse_tmp
        i = 0
        f = open(sfsFolder, "w")
        for i in range(len(files)):
            f.write(files[i] + "\n")
        f.close()
        time.sleep(3)

def get_data(url):
    """Download json data from manamyzer url"""
    r = requests.get(url)
    return r.json()

def feature_extraction(data):
    """Extract the features from manalyzer data"""
    features = {}
    md5 = data.keys()[0]
    data = data[md5]
    features['md5'] = md5
    features['Machine']= MACHINE_TYPES[data['PE Header']['Machine']]
    features['SizeOfOptionalHeader'] = data['PE Header']['SizeOfOptionalHeader']
    features['Characteristics'] = 0
    for charac in data['PE Header']['Characteristics']:
        features['Characteristics'] += PE_CHARACTERISTICS[charac]
    features['SizeOfCode'] = data['Image Optional Header']['SizeOfCode']
    features['SizeOfInitializedData']         =         data['Image         Optional
Header']['SizeOfInitializedData']
    features['SizeOfUninitializedData']        =        data['Image        Optional
Header']['SizeOfUninitializedData']
    features['AddressOfEntryPoint']         =         data['Image         Optional
Header']['AddressOfEntryPoint']
    features['BaseOfCode'] = data['Image Optional Header']['AddressOfEntryPoint']
    try:
        features['BaseOfData'] = data['Image Optional Header']['BaseOfData']
    except KeyError:
        features['BaseOfData'] = 0
    features['ImageBase'] = data['Image Optional Header']['ImageBase']
    features['SectionAlignment']          =          data['Image          Optional
Header']['SectionAlignment']
    features['FileAlignment'] = data['Image Optional Header']['FileAlignment']
    osv = data['Image Optional Header']['OperatingSystemVersion'].split('.')
    features['MajorOperatingSystemVersion'] = int(osv[0])
    features['MinorOperatingSystemVersion'] = int(osv[1])
    ssv = data['Image Optional Header']['SubsystemVersion'].split('.')
```

```
    features['MajorSubsystemVersion'] = int(ssv[0])
    features['MinorSubsystemVersion'] = int(ssv[1])
    features['Subsystem'] = SUBSYSTEMS[data['Image Optional Header']['Subsystem']]
    features['DllCharacteristics'] = 0
    for char in data["Image Optional Header"]["DllCharacteristics"]:
        features['DllCharacteristics'] += DLL_CHARACTERISTICS[char]
    features['SizeOfStackReserve'] = data['Image Optional
Header']['SizeofStackReserve']
    features['SizeOfStackCommit'] = data['Image Optional
Header']['SizeofStackCommit']
    features['SizeOfHeapReserve'] = data['Image Optional
Header']['SizeofHeapReserve']
    features['SizeOfHeapCommit'] = data['Image Optional
Header']['SizeofHeapCommit']
    features['LoaderFlags'] = data['Image Optional Header']['LoaderFlags']
    features['NumberOfRvaAndSizes'] = data['Image Optional
Header']['NumberOfRvaAndSizes']


    features['SectionsNb'] = len(data['Sections'])
    entropy = map(lambda x:x['Entropy'], data['Sections'].values())
    features['SectionsMeanEntropy'] = sum(entropy) / float(len(entropy))
    features['SectionsMinEntropy'] = min(entropy)
    features['SectionsMaxEntropy'] = max(entropy)
    raw_sizes = map(lambda x:x['SizeOfRawData'], data['Sections'].values())
    features['SectionsMeanRawsize'] = sum(raw_sizes) / float(len(raw_sizes))
    features['SectionsMinRawsize'] = min(raw_sizes)
    features['SectionsMaxRawsize'] = max(raw_sizes)
    virtual_sizes = map(lambda x:x['VirtualSize'], data['Sections'].values())
    features['SectionsMeanVirtualsize'] = sum(virtual_sizes) /
float(len(virtual_sizes))
    features['SectionsMinVirtualsize'] = min(virtual_sizes)
    features['SectionsMaxVirtualsize'] = max(virtual_sizes)


    if 'Imports' in data.keys():
        features['ImportsNbDLL'] = len(data['Imports'])
        features['ImportsNb'] = sum(map(len, data['Imports'].values()))
    else:
        features['ImportsNbDLL'] = 0
        features['ImportsNb'] = 0


    if 'Resources' in data.keys():
        features['ResourcesNb'] = len(data['Resources'])
        entropy = map(lambda x:x['Entropy'], data['Resources'].values())
        features['ResourcesMeanEntropy'] = sum(entropy) / float(len(entropy))
        features['ResourcesMinEntropy'] = min(entropy)
        features['ResourcesMaxEntropy'] = max(entropy)
        sizes = map(lambda x:x['Size'], data['Resources'].values())
        features['ResourcesMeanSize'] = sum(sizes) / float(len(sizes))
```

```
        features['ResourcesMinSize'] = min(sizes)
        features['ResourcesMaxSize'] = max(sizes)
    else:
        features['ResourcesNb'] = 0
        features['ResourcesMeanEntropy'] = 0
        features['ResourcesMinEntropy'] = 0
        features['ResourcesMaxEntropy'] = 0
        features['ResourcesMeanSize'] = 0
        features['ResourcesMinSize'] = 0
        features['ResourcesMaxSize'] = 0

    if "Version Info" in data.keys():
        features['VersionInformationSize'] = len(data['Version Info'].keys())
    else:
        features['VersionInformationSize'] = 0

    return features


if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Detect malicious file from
manalyzer infos')
    parser.add_argument('URL', help='Manalyzer url')
    args = parser.parse_args()


    clf = joblib.load(os.path.join(
        os.path.dirname(os.path.realpath(__file__)),
        'classifier/classifier.pkl'
    ))
    features = pickle.loads(open(os.path.join(
        os.path.dirname(os.path.realpath(__file__)),
        'classifier/features.pkl'),
        'r').read()
    )

    if 'manalyzer.org' not in args.URL:
        print('This is not a manalyzer url')
        sys.exit(1)
    if '/report/' in args.URL:
        url = args.URL.replace('/report/', '/json/')
    else:
        url = args.URL

    data = get_data(url)
    if data == {}:
        print("Impossible to retrieve the data, quitting")
        sys.exit(1)
    else:
```

```
        data_pe = feature_extraction(data)
        pe_features = map(lambda x:data_pe[x], features)
        res= clf.predict([pe_features])[0]
        print('The file %s is %s' % (
            data_pe['md5'],
            ['malicious', 'legitimate'][res])
        )


data = pd.read_csv('data.csv', sep='|')
X = data.drop(['Name', 'md5', 'legitimate'], axis=1).values
y = data['legitimate'].values

print('Researching important feature based on %i total features\n' % X.shape[1])


fsel = ske.ExtraTreesClassifier().fit(X, y)
model = SelectFromModel(fsel, prefit=True)
X_new = model.transform(X)
nb_features = X_new.shape[1]

X_train, X_test, y_train, y_test = cross_validation.train_test_split(X_new, y
,test_size=0.2)

features = []

print('%i features identified as important:' % nb_features)

indices = np.argsort(fsel.feature_importances_)[::-1][:nb_features]
for f in range(nb_features):
    print("%d. feature %s (%f)" % (f + 1, data.columns[2+indices[f]],
fsel.feature_importances_[indices[f]]))


for f in sorted(np.argsort(fsel.feature_importances_)[::-1][:nb_features]):
    features.append(data.columns[2+f])


algorithms = {
        "DecisionTree": tree.DecisionTreeClassifier(max_depth=10),
        "RandomForest": ske.RandomForestClassifier(n_estimators=50),
        "GradientBoosting": ske.GradientBoostingClassifier(n_estimators=50),
        "AdaBoost": ske.AdaBoostClassifier(n_estimators=100),
        "GNB": GaussianNB()
    }

results = {}
print("\nNow testing algorithms")
for algo in algorithms:
    clf = algorithms[algo]
    clf.fit(X_train, y_train)
```

```
    score = clf.score(X_test, y_test)
    print("%s : %f %%" % (algo, score*100))
    results[algo] = score

winner = max(results, key=results.get)
print('\nWinner algorithm is %s with a %f %% success' % (winner,
results[winner]*100))


print('Saving algorithm and feature list in classifier directory...')
joblib.dump(algorithms[winner], 'classifier/classifier.pkl')
open('classifier/features.pkl', 'wb').write(pickle.dumps(features))
print('Saved')


clf = algorithms[winner]
res = clf.predict(X_test)
mt = confusion_matrix(y_test, res)
print("False positive rate : %f %%" % ((mt[0][1] / float(sum(mt[0])))*100))
print('False negative rate : %f %%' % ( (mt[1][0] / float(sum(mt[1])))*100)))

epochs = 100
nclass = 12

def loadDataset():

    filename='D:/02-14-2018.csv/02-14-2018.csv'

    trainfile = pd.read_csv(filename)
    data = pd.DataFrame(trainfile).to_numpy()
    data=data[data[:,67]!='DrDoS_LDAP']
    np.random.shuffle(data)

    label = data[:, 67].astype('str')

    label[label == 'WebDDoS']       = 0
    label[label == 'BENIGN']        = 1
    label[label == 'UDP-lag']       = 2
    label[label == 'DrDoS_NTP']     = 3
    label[label == 'Syn']           = 4
    label[label == 'DrDoS_SSDP']    = 5
    label[label == 'DrDoS_UDP']     = 6
    label[label == 'DrDoS_NetBIOS'] = 7
    label[label == 'DrDoS_MSSQL']   = 8
    label[label == 'DrDoS_SNMP']    = 9
    label[label == 'TFTP']          = 10
    label[label == 'DrDoS_DNS']     = 11
```

```
    inx_sel=-1+np.array([38,47,37,48,11,9,7,52,10,36,1,34,4,17,19,57,21,
                         18,22,24,32,50,23,55,51,5,3,39,40,43,58,12,25,
                         20,2,35,67,33,6,53])


    data=data[:,inx_sel]
    dmin = data.min(axis=0)
    dmax = data.max(axis=0)
    data=(data-dmin)/(dmax-dmin)
     data = np.log(data-dmin+1.0)



    train_data, test_data, train_label, test_label = \
        train_test_split(data, label, test_size=0.20, stratify=label)


    train_data, val_data, train_label, val_label = \
        train_test_split(train_data,                    train_label,test_size=0.125,
stratify=train_label)


    return train_data.astype('float32'), train_label.astype('int32'), \
        val_data.astype('float32'), val_label.astype('int32'), \
            test_data.astype('float32'), test_label.astype('int32')


train_data,   train_labelp,   val_data,   val_labelp,   test_data,   test_labelp   =
loadDataset()


train_label = to_categorical(train_labelp, nclass)
val_label   = to_categorical(val_labelp,   nclass)
test_label  = to_categorical(test_labelp,  nclass)

print('train_data.shape=', train_data.shape)
print('test_data.shape=',  test_data.shape)
print('test_data.shape=',  val_data.shape)


inshape=train_data.shape[1]


class_weights = class_weight.compute_class_weight(class_weight='balanced',
                                                  classes=np.unique(
                                                      train_labelp),
                                                  y=train_labelp)


class_weights = {i: class_weights[i] for i in range(len(class_weights))}
```

```
earlyStopping = EarlyStopping(monitor='val_loss',
                              patience=30,
                              verbose=0,
                              mode='min')

modelCheckPoint   =   ModelCheckpoint('./savemodels/model5class.weights.{epoch:03d}-
{val_acc:.4f}.hdf5',
                                    save_best_only=True,
                                    monitor='val_acc',
                                    mode='max')




history = model.fit(train_data,
                    train_label,
                    shuffle=True,
                    epochs=epochs,
                    batch_size=256,  # 256,#128,#32, 64
                    # validation_data=validation_generator,
                    # validation_split=0.2,
                    # validation_data=(val_data,val_label),
                    validation_data=(val_data, val_label),
                    callbacks=[modelCheckPoint],
                    class_weight=class_weights,
                    workers=3)


str_models = os.listdir('./savemodels')
str_models = np.sort(str_models)
best_model = str_models[str_models.size-1]
print('best_model=', best_model)
model.load_weights('./savemodels/'+best_model)


print('TEST DATA-Confusion matrix:')
pred = model.predict(test_data)
pred_y = pred.argmax(axis=-1)

cm = confusion_matrix(test_labelp.astype('int32'), pred_y)
print(cm)

print('Accuracy ratios for each class')
print('WebDDoS      =', cm[0, 0]/np.sum(cm[0, :]))
print('BENIGN       =', cm[1, 1]/np.sum(cm[1, :]))
print('UDP-lag      =', cm[2, 2]/np.sum(cm[2, :]))
print('DrDoS_NTP    =', cm[3, 3]/np.sum(cm[3, :]))
print('Syn          =', cm[4, 4]/np.sum(cm[4, :]))
```

```
print('DrDoS_SSDP   =', cm[5, 5]/np.sum(cm[5, :]))
print('DrDoS_UDP    =', cm[6, 6]/np.sum(cm[6, :]))
print('DrDoS_NetBIOS=', cm[7, 7]/np.sum(cm[7, :]))
print('DrDoS_MSSQL  =', cm[8, 8]/np.sum(cm[8, :]))
print('DrDoS_SNMP   =', cm[9, 9]/np.sum(cm[9, :]))
print('TFTP         =', cm[10,10]/np.sum(cm[10, :]))
print('DrDoS_DNS    =', cm[11,11]/np.sum(cm[11, :]))



from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
label=np.array(["WebDDoS","BENIGN","UDP-lag","DrDoS_NTP","Syn ",
                "DrDoS_SSDP","DrDoS_UDP","DrDoS_NetBIOS","DrDoS_MSSQL",
                "DrDoS_SNMP","TFTP","DrDoS_DNS"])

cmo = ConfusionMatrixDisplay(cm,display_labels=label)
fig, ax = plt.subplots(figsize=(12,12))
cmo.plot(ax=ax, xticks_rotation=45)



acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

np.save('historydata.npy',[acc,val_acc,loss,val_loss])
[acc, val_acc, loss, val_loss] = np.load('historydata.npy')

plt.figure()
epochs = range(len(acc))
plt.plot(epochs, acc, 'b', label='Training acc')
plt.plot(epochs, val_acc, 'r.', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')

plt.legend()
plt.figure()
plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r.', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

def model_lstm(lr=1e-4,N=64,inshape=40,nclass=12):
    in1=Input(shape=(inshape,1))
    x=layers.LSTM(N, activation='tanh')(in1)
    x=Dropout(0.1)(x)
```

```
    x=Dense(128, activation='relu')(x)
    out1=Dense(nclass, activation='softmax')(x)
    model=Model(inputs=in1,outputs=out1)

    model.compile(optimizer=optimizers.Adam(lr),
              loss=losses.categorical_crossentropy,
              metrics=['acc'])
    return model



def model_conv1D(lr=1e-4,N=64,inshape=40,nclass=12):
    in1=Input(shape=(inshape,1))
    x=Conv1D(N, 3,padding='same',activation='relu')(in1)
    x=Conv1D(N, 3,padding='same',activation='relu')(x)

    x=layers.Flatten()(x)
    x=Dropout(0.1)(x)
    x=Dense(128, activation='relu')(x)
    out1=Dense(nclass, activation='softmax')(x)
    model=Model(inputs=in1,outputs=out1)

    model.compile(optimizer=optimizers.Adam(lr),
              loss=losses.categorical_crossentropy,
              metrics=['acc'])
    return model




def model_dense(lr=1e-4,N=64,inshape=40,nclass=12):
    in1=Input(shape=(inshape,1))
    x=Dense(N,activation='relu')(in1)
    x=Dense(N,activation='relu')(x)
    x=layers.Flatten()(x)
    x=Dropout(0.1)(x)
    x=Dense(128, activation='relu')(x)
    out1=Dense(nclass, activation='softmax')(x)
    model=Model(inputs=in1,outputs=out1)

    model.compile(optimizer=optimizers.Adam(lr),
              loss=losses.categorical_crossentropy,
              metrics=['acc'])
    return model


def model_conv1D_large(nfeat=40,lr=1e-2,nclass=12):
    in1=Input(shape=(nfeat,1))
    x=Conv1D(64, 3,padding='same' ,activation='relu')(in1)
    x=Conv1D(64, 3,padding='same' ,activation='relu')(x)
```

```
    x=AvgPool1D()(x)
    x=Conv1D(128,5,padding='same' ,activation='relu')(x)
    x=AvgPool1D()(x)
    x=Conv1D(256,7,padding='same'  ,activation='relu')(x)
    x=AvgPool1D()(x)
    x=Conv1D(512,9,padding='same' ,activation='relu')(x)
    x=Flatten()(x)
    x=Dropout(0.4)(x)
    x=Dense(512, activation='relu')(x)
    output=Dense(nclass, activation='softmax')(x)
    model = Model(inputs=in1, outputs=output)


    opt=optimizers.Adam(lr)

    model.compile(optimizer=opt,#Adam(lr=1e-2),
              loss=losses.categorical_crossentropy,
              metrics=['acc'])
    return model

def model_conv1D_binary(nfeat=32,lr=1e-2,nclass=12):
    in1=Input(shape=(nfeat,1))
    x=Conv1D(64, 3,padding='same' ,activation='relu')(in1)
    x=Conv1D(64, 3,padding='same' ,activation='relu')(x)
    x=AvgPool1D()(x)
    x=Conv1D(128,5,padding='same' ,activation='relu')(x)
    x=AvgPool1D()(x)
    x=Conv1D(256,7,padding='same'  ,activation='relu')(x)
    x=AvgPool1D()(x)
    x=Conv1D(512,9,padding='same' ,activation='relu')(x)
    x=Flatten()(x)
    x=Dropout(0.2)(x)
    x=Dense(512, activation='relu')(x)
    output=Dense(nclass, activation='sigmoid')(x)
    model = Model(inputs=in1, outputs=output)


    opt=optimizers.RMSprop(lr)

    model.compile(optimizer=opt,#Adam(lr=1e-2),
              loss=losses.binary_crossentropy,
              metrics=['acc'])
    return model

import numpy as np
import matplotlib
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import PySimpleGUI as sg
```

```
def scanids():
    layout = [
        [sg.Canvas(size=(150, 150), background_color='red', key='canvas')],
        [sg.Text('Change circle color to:'), sg.Button('Red'), sg.Button('Blue')]
    ]

    window = sg.Window('Сканування', layout, finalize=True)

    cir = window['canvas'].TKCanvas.create_oval(50, 50, 100, 100)

    while True:
        event, values = window.read()
        if event == sg.WIN_CLOSED:
            break
        if event in ('Blue', 'Red'):
            window['canvas'].TKCanvas.itemconfig(cir, fill=event)

    window.close()


def main():
    sg.theme('Material2')
    sg.theme('black')


    menu_def = [['&Файл', ['&Open     Ctrl-O', '&Save       Ctrl-S', '&Properties',
'E&xit']],
                ['&Редагування',   ['Edit   Me',   'Special',   'Normal',['Normal1',
'Normal2'] , 'Undo']],
                ['&IDS', ['Special', 'Normal',['Normal1', 'Normal2'], 'Undo']],
                ['&Параметри', ['---', 'Command &1::Command_Key', 'Command &2', '--
-', 'Command &3', 'Command &4']],
                ['&Довідка', ['&About...']], ]

    layout = [[sg.MenubarCustom(menu_def, pad=(0,0), k='-CUST MENUBAR-')],
              [sg.Multiline(size=(70,  40),   reroute_cprint=True,  write_only=True,
no_scrollbar=True, k='-MLINE-')],
              [sg.Button('Виявлення вторгнень'), sg.Button('Вихід')]]

    window      =      sg.Window("Система      виявлення      вторгнень",      layout,
use_custom_titlebar=True,                                          keep_on_top=True,
right_click_menu=sg.MENU_RIGHT_CLICK_EDITME_VER_EXIT)

    # ------ Event Loop ------ #
    while True:
        event, values = window.read()
        # convert ButtonMenu event so they look like Menu events

        if event in (sg.WIN_CLOSED, 'Exit'):
            break
```

```
        sg.cprint(f'event      =      {event}',      c=(sg.theme_background_color(),
sg.theme_text_color()))
        sg.cprint(f'values       =       {values}',c=(sg.theme_input_text_color(),
sg.theme_input_background_color()))

        # ------ Process menu choices ------ #
        if event == 'About...':
            window.disappear()
            sg.popup('Про ПЗ', 'Система виявлення вторгнень',
                     'Розроблена студентом ЦНТУ', 'Шовкопляс Ю.С.', 'КІ-21М', '2022
рік',   grab_anywhere=True, keep_on_top=True)
            window.reappear()
        elif event == 'Edit Me':
            sg.execute_editor(__file__)
        elif event == 'Version':
            sg.popup_scrolled(__file__,     sg.get_versions(),     keep_on_top=True,
non_blocking=True)
        elif event.startswith('Open'):
            filename = sg.popup_get_file('file to open', no_window=True)
            print(filename)

    while True:
        if event == sg.WIN_CLOSED:
            break
        elif event == 'Виявлення вторгнень':
            scanids()

    window.close()


if __name__ == '__main__':
    main()

KDC_ERR_ETYPE_NOSUPP = 14
KDC_ERR_PREAUTH_REQUIRED = 25

NT_UNKNOWN = 0
NT_PRINCIPAL = 1
NT_SRV_INST = 2
NT_SRV_HST = 3
NT_SRV_XHST = 4
NT_UID = 5
NT_X500_PRINCIPAL = 6
NT_SMTP_NAME = 7
NT_ENTERPRISE = 10


AD_IF_RELEVANT = 1
AD_WIN2K_PAC = 128

socket_m.setdefaulttimeout(3)
```

```
def _c(n, t):
    return t.clone(tagSet=t.tagSet + Tag(tagClassContext, tagFormatSimple, n))

def _v(n, t):
    return  t.clone(tagSet=t.tagSet  +  Tag(tagClassContext,  tagFormatSimple,  n),
cloneValueFlag=True)

def application(n):
    return Sequence.tagSet + Tag(tagClassApplication, tagFormatSimple, n)

class Microseconds(Integer): pass

class KerberosString(GeneralString): pass

class Realm(KerberosString): pass

class PrincipalName(Sequence):
    componentType = NamedTypes(
        NamedType('name-type', _c(0, Integer())),
        NamedType('name-string',                                                _c(1,
SequenceOf(componentType=KerberosString()))))

class KerberosTime(GeneralizedTime): pass

class HostAddress(Sequence):
    componentType = NamedTypes(
        NamedType('addr-type', _c(0, Integer())),
        NamedType('address', _c(1, OctetString())))

class HostAddresses(SequenceOf):
    componentType = HostAddress()

class AuthorizationData(SequenceOf):
    componentType = Sequence(componentType=NamedTypes(
            NamedType('ad-type', _c(0, Integer())),
            NamedType('ad-data', _c(1, OctetString()))))

class PAData(Sequence):
    componentType = NamedTypes(
        NamedType('padata-type', _c(1, Integer())),
        NamedType('padata-value', _c(2, OctetString())))

class KerberosFlags(BitString): pass

class EncryptedData(Sequence):
    componentType = NamedTypes(
        NamedType('etype', _c(0, Integer())),
        OptionalNamedType('kvno', _c(1, Integer())),
        NamedType('cipher', _c(2, OctetString())))
```

```
class EncryptionKey(Sequence):
    componentType = NamedTypes(
        NamedType('keytype', _c(0, Integer())),
        NamedType('keyvalue', _c(1, OctetString())))

class CheckSum(Sequence):
    componentType = NamedTypes(
        NamedType('cksumtype', _c(0, Integer())),
        NamedType('checksum', _c(1, OctetString())))

class Ticket(Sequence):
    tagSet = application(1)
    componentType = NamedTypes(
        NamedType('tkt-vno', _c(0, Integer())),
        NamedType('realm', _c(1, Realm())),
        NamedType('sname', _c(2, PrincipalName())),
        NamedType('enc-part', _c(3, EncryptedData())))

class APOptions(KerberosFlags): pass

class APReq(Sequence):
    tagSet = application(14)
    componentType = NamedTypes(
        NamedType('pvno', _c(0, Integer())),
        NamedType('msg-type', _c(1, Integer())),
        NamedType('ap-options', _c(2, APOptions())),
        NamedType('ticket', _c(3, Ticket())),
        NamedType('authenticator', _c(4, EncryptedData())))

class Authenticator(Sequence):
    tagSet = application(2)
    componentType = NamedTypes(
        NamedType('authenticator-vno', _c(0, Integer())),
        NamedType('crealm', _c(1, Realm())),
        NamedType('cname', _c(2, PrincipalName())),
        OptionalNamedType('cksum', _c(3, CheckSum())),
        NamedType('cusec', _c(4, Microseconds())),
        NamedType('ctime', _c(5, KerberosTime())),
        OptionalNamedType('subkey', _c(6, EncryptionKey())),
        OptionalNamedType('seq-number', _c(7, Integer())),
        OptionalNamedType('authorization-data', _c(8, AuthorizationData())))

class KDCOptions(KerberosFlags): pass

class KdcReqBody(Sequence):
    componentType = NamedTypes(
        NamedType('kdc-options', _c(0, KDCOptions())),
        OptionalNamedType('cname', _c(1, PrincipalName())),
        NamedType('realm', _c(2, Realm())),
        OptionalNamedType('sname', _c(3, PrincipalName())),
        OptionalNamedType('from', _c(4, KerberosTime())),
```

```
        NamedType('till', _c(5, KerberosTime())),
        OptionalNamedType('rtime', _c(6, KerberosTime())),
        NamedType('nonce', _c(7, Integer())),
        NamedType('etype', _c(8, SequenceOf(componentType=Integer()))),
        OptionalNamedType('addresses', _c(9, HostAddresses())),
        OptionalNamedType('enc-authorization-data', _c(10, EncryptedData())),
        OptionalNamedType('additional-tickets',                                    _c(11,
SequenceOf(componentType=Ticket()))))

class KdcReq(Sequence):
    componentType = NamedTypes(
        NamedType('pvno', _c(1, Integer())),
        NamedType('msg-type', _c(2, Integer())),
        NamedType('padata', _c(3, SequenceOf(componentType=PAData()))),
        NamedType('req-body', _c(4, KdcReqBody())))

class TicketFlags(KerberosFlags): pass

class KrbError(Sequence):
    tagSet = application(30)
    componentType = NamedTypes(
    NamedType('pvno', _c(0, Integer())),
    NamedType('msg-type', _c(1, Integer())),
    OptionalNamedType('ctime', _c(2, KerberosTime())),
    OptionalNamedType('cusec', _c(3, Microseconds())),
    NamedType('stime', _c(4, KerberosTime())),
    NamedType('susec', _c(5, Microseconds())),
    NamedType('error-code', _c(6, Integer())),
    OptionalNamedType('crealm', _c(7, Realm())),
    OptionalNamedType('cname', _c(8, PrincipalName())),
    NamedType('realm', _c(9, Realm())),
    NamedType('sname', _c(10, PrincipalName())),
    OptionalNamedType('e-text', _c(11, KerberosString())),
    OptionalNamedType('e-data ', _c(12, OctetString())))


class AsReq(KdcReq):
    tagSet = application(10)

class TgsReq(KdcReq):
    tagSet = application(12)

class KdcRep(Sequence):
    componentType = NamedTypes(
        NamedType('pvno', _c(0, Integer())),
        NamedType('msg-type', _c(1, Integer())),
        OptionalNamedType('padata', _c(2, SequenceOf(componentType=PAData()))),
        NamedType('crealm', _c(3, Realm())),
        NamedType('cname', _c(4, PrincipalName())),
        NamedType('ticket', _c(5, Ticket())),
        NamedType('enc-part', _c(6, EncryptedData())))
```

```
class AsRep(KdcRep):
    tagSet = application(11)

class TgsRep(KdcRep):
    tagSet = application(13)

class LastReq(SequenceOf):
    componentType = Sequence(componentType=NamedTypes(
            NamedType('lr-type', _c(0, Integer())),
            NamedType('lr-value', _c(1, KerberosTime()))))

class PaEncTimestamp(EncryptedData): pass

class PaEncTsEnc(Sequence):
    componentType = NamedTypes(
        NamedType('patimestamp', _c(0, KerberosTime())),
        NamedType('pausec', _c(1, Microseconds())))

class EncKDCRepPart(Sequence):
    componentType = NamedTypes(
        NamedType('key', _c(0, EncryptionKey())),
        NamedType('last-req', _c(1, LastReq())),
        NamedType('nonce', _c(2, Integer())),
        OptionalNamedType('key-expiration', _c(3, KerberosTime())),
        NamedType('flags', _c(4, TicketFlags())),
        NamedType('authtime', _c(5, KerberosTime())),
        OptionalNamedType('starttime', _c(6, KerberosTime())),
        NamedType('endtime', _c(7, KerberosTime())),
        OptionalNamedType('renew-till', _c(8, KerberosTime())),
        NamedType('srealm', _c(9, Realm())),
        NamedType('sname', _c(10, PrincipalName())),
        OptionalNamedType('caddr', _c(11, HostAddresses())),
        OptionalNamedType('encrypted-pa-data',                                  _c(12,
SequenceOf(componentType=PAData()))))

class EncASRepPart(EncKDCRepPart):
    tagSet = application(25)

class EncTGSRepPart(EncKDCRepPart):
    tagSet = application(26)

class TransitedEncoding(Sequence):
    componentType = NamedTypes(
        NamedType('tr-type', _c(0, Integer())),
        NamedType('contents', _c(1, OctetString())))

class EncTicketPart(Sequence):
    tagSet = application(3)
    componentType = NamedTypes(
        NamedType('flags', _c(0, TicketFlags())),
```

```
        NamedType('key', _c(1, EncryptionKey())),
        NamedType('crealm', _c(2, Realm())),
        NamedType('cname', _c(3, PrincipalName())),
        NamedType('transited', _c(4, TransitedEncoding())),
        NamedType('authtime', _c(5, KerberosTime())),
        OptionalNamedType('starttime', _c(6, KerberosTime())),
        NamedType('endtime', _c(7, KerberosTime())),
        OptionalNamedType('renew-till', _c(8, KerberosTime())),
        OptionalNamedType('caddr', _c(9, HostAddresses())),
        OptionalNamedType('authorization-data', _c(10, AuthorizationData())))

class KerbPaPacRequest(Sequence):
    componentType = NamedTypes(
        NamedType('include-pac', _c(0, Boolean())))

def build_req_body(realm, service, host, nonce, cname=None,
authorization_data=None, etype=RC4_HMAC):
    req_body = KdcReqBody()

    # (Forwardable, Proxiable, Renewable, Canonicalize)
    if service == 'krbtgt':

        # 使用了一个 unused 18 的标志位，用来区分请求，0x50802000
        req_body['kdc-options'] = "'01010000100000000010000000000000'B"
    else:  # other services (Forwardable, Renewable, Canonicalize)
        req_body['kdc-options'] = "'01000000100000010000000000000000'B"

    if cname is not None:
        req_body['cname'] = None
        req_body['cname']['name-type'] = NT_PRINCIPAL
        req_body['cname']['name-string'] = None
        req_body['cname']['name-string'][0] = cname

    req_body['realm'] = realm

    req_body['sname'] = None
    req_body['sname']['name-type'] = NT_SRV_INST
    req_body['sname']['name-string'] = None
    req_body['sname']['name-string'][0] = service
    req_body['sname']['name-string'][1] = host

    req_body['from'] = '19700101000000Z'
    req_body['till'] = '19700101000000Z'
    req_body['rtime'] = '19700101000000Z'
    req_body['nonce'] = nonce

    req_body['etype'] = None
    req_body['etype'][0] = etype

    if authorization_data is not None:
```

```
        req_body['enc-authorization-data'] = None
        req_body['enc-authorization-data']['etype'] = authorization_data[0]
        req_body['enc-authorization-data']['cipher'] = authorization_data[1]

    return req_body

def    build_authenticator(realm,    name,    chksum,    subkey,    current_time,
authorization_data=None):
    auth = Authenticator()

    auth['authenticator-vno'] = 5

    auth['crealm'] = realm

    auth['cname'] = None
    auth['cname']['name-type'] = NT_PRINCIPAL
    auth['cname']['name-string'] = None
    auth['cname']['name-string'][0] = name

    auth['cksum'] = None
    auth['cksum']['cksumtype'] = chksum[0]
    auth['cksum']['checksum'] = chksum[1]

    gt, ms = epoch2gt(current_time, microseconds=True)
    auth['cusec'] = ms
    auth['ctime'] = gt

    auth['subkey'] = None
    auth['subkey']['keytype'] = subkey[0]
    auth['subkey']['keyvalue'] = subkey[1]

    if authorization_data is not None:
        auth['authorization-data'] = _v(8, authorization_data)

    return auth

def build_ap_req(ticket, key, msg_type, authenticator):
    enc_auth = encrypt(key[0], key[1], msg_type, encode(authenticator))

    ap_req = APReq()
    ap_req['pvno'] = 5
    ap_req['msg-type'] = 14
    ap_req['ap-options'] = "'00000000000000000000000000000000'B"
    ap_req['ticket'] = _v(3, ticket)

    ap_req['authenticator'] = None
    ap_req['authenticator']['etype'] = key[0]
    ap_req['authenticator']['cipher'] = enc_auth

    return ap_req
```

```
def build_tgs_req(target_realm, target_service, target_host,
                  user_realm, user_name, tgt, session_key, subkey,
                  nonce, current_time, authorization_data=None, pac_request=None,
etype=RC4_HMAC):

    if authorization_data is not None:
        ad1 = AuthorizationData()
        ad1[0] = None
        ad1[0]['ad-type'] = authorization_data[0]
        ad1[0]['ad-data'] = authorization_data[1]
        ad = AuthorizationData()
        ad[0] = None
        ad[0]['ad-type'] = AD_IF_RELEVANT
        ad[0]['ad-data'] = encode(ad1)
        enc_ad = (subkey[0], encrypt(subkey[0], subkey[1], 5, encode(ad)))
    else:
        ad = None
        enc_ad = None

    req_body = build_req_body(target_realm, target_service, target_host, nonce,
authorization_data=enc_ad, etype=etype)
    chksum = (RSA_MD5, checksum(RSA_MD5, encode(req_body)))

    authenticator = build_authenticator(user_realm, user_name, chksum, subkey,
current_time)#, ad)
    ap_req = build_ap_req(tgt, session_key, 7, authenticator)

    tgs_req = TgsReq()
    tgs_req['pvno'] = 5
    tgs_req['msg-type'] = 12

    tgs_req['padata'] = None
    tgs_req['padata'][0] = None
    tgs_req['padata'][0]['padata-type'] = 1
    tgs_req['padata'][0]['padata-value'] = encode(ap_req)

    if pac_request is not None:
        pa_pac_request = KerbPaPacRequest()
        pa_pac_request['include-pac'] = pac_request
        tgs_req['padata'][1] = None
        tgs_req['padata'][1]['padata-type'] = 128
        tgs_req['padata'][1]['padata-value'] = encode(pa_pac_request)

    tgs_req['req-body'] = _v(4, req_body)

    return tgs_req

def build_pa_enc_timestamp(current_time, key):
    gt, ms = epoch2gt(current_time, microseconds=True)
    pa_ts_enc = PaEncTsEnc()
    pa_ts_enc['patimestamp'] = gt
```

```
    pa_ts_enc['pausec'] = ms

    pa_ts = PaEncTimestamp()
    pa_ts['etype'] = key[0]
    pa_ts['cipher'] = encrypt(key[0], key[1], 1, encode(pa_ts_enc))

    return pa_ts

def build_as_req(target_realm, user_name, key, current_time, nonce, pac_request,
etype = RC4_HMAC):
    req_body = build_req_body(target_realm, 'krbtgt', target_realm, nonce,
user_name, None, etype)

    as_req = AsReq()

    as_req['pvno'] = 5
    as_req['msg-type'] = 10

    as_req['padata'] = None
    pa_counter = 0
    if key is not None:
        pa_ts = build_pa_enc_timestamp(current_time, key)
        as_req['padata'][pa_counter] = None
        as_req['padata'][pa_counter]['padata-type'] = 2
        as_req['padata'][pa_counter]['padata-value'] = encode(pa_ts)
        pa_counter = pa_counter + 1

    if pac_request is not None:
        pa_pac_request = KerbPaPacRequest()
        pa_pac_request['include-pac'] = pac_request
        as_req['padata'][pa_counter] = None
        as_req['padata'][pa_counter]['padata-type'] = 128
        as_req['padata'][pa_counter]['padata-value'] = encode(pa_pac_request)
        pa_counter = pa_counter + 1

    as_req['req-body'] = _v(4, req_body)

    return as_req

def send_req(req, kdc, port=88):
    data = encode(req)
    data = pack('>I', len(data)) + data
    sock = socket()
    sock.connect((kdc, port))
    sock.send(data)
    return sock

def recv_rep(sock):
    data = bytes('', encoding="utf-8")
    datalen = None
    while True:
```

```
        rep = sock.recv(8192)
        if not rep:
            sock.close()
            raise IOError('Connection error')
        assert isinstance(rep, bytes)
        data += rep
        if len(rep) >= 4:
            if datalen is None:
                datalen = unpack('>I', rep[:4])[0]
            if len(data) >= 4 + datalen:
                sock.close()
                return data[4:4 + datalen]

class CCache(object):
    def __init__(self, primary_principal, credentials=[], header=DEFAULT_HEADER):
        if not isinstance(primary_principal, CCachePrincipal):
            if isinstance(primary_principal, str) and '@' in primary_principal:
                realm, user_name = primary_principal.split('@', 1)
            elif isinstance(primary_principal, tuple) and len(primary_principal) ==
2:
                realm, user_name = primary_principal
            else:
                raise   ValueError('Bad   primary   principal   format:   %r'   %
primary_principal)
            primary_principal = CCachePrincipal(NT_PRINCIPAL, realm, [user_name])

        self.primary_principal = primary_principal
        self.credentials = credentials
        self.header = header

    @classmethod
    def load(cls, filename):
        fp = open(filename, 'rb')
        version, headerlen = unpack('>HH', fp.read(4))
        if version != VERSION:
            raise ValueError('Unsupported version: 0x%04x' % version)
        header = fp.read(headerlen)
        primary_principal = cls.read_principal(fp)
        credentials = []
        while True:
            try:
                credentials.append(cls.read_credential(fp))
            except struct.error:
                break
        fp.close()
        return cls(primary_principal, credentials, header)

    def save(self, filename):
        fp = open(filename, 'wb')
        fp.write(pack('>HH', VERSION, len(self.header)))
        fp.write(self.header)
```

```
        self.write_principal(fp, self.primary_principal)
        for cred in self.credentials:
            self.write_credential(fp, cred)
        fp.close()

    def add_credential(self, newcred):
        for i in range(len(self.credentials)):
            if self.credentials[i].client == newcred.client and \
                    self.credentials[i].server == newcred.server:
                self.credentials[i] = newcred
                return
        self.credentials.append(newcred)

    @classmethod
    def read_string(cls, fp):
        length = unpack('>I', fp.read(4))[0]
        return fp.read(length)

    @classmethod
    def write_string(cls, fp, s):
        fp.write(pack('>I', len(s)))
        fp.write(s)

    @classmethod
    def read_principal(cls, fp):
        name_type, num_components = unpack('>II', fp.read(8))
        realm = cls.read_string(fp)
        components = [cls.read_string(fp) for i in range(num_components)]
        return CCachePrincipal(name_type, realm, components)

    @classmethod
    def write_principal(cls, fp, p):
        fp.write(pack('>II', p.name_type, len(p.components)))
        cls.write_string(fp, p.realm)
        for comp in p.components:
            cls.write_string(fp, comp)

    @classmethod
    def read_keyblock(cls, fp):
        keytype, etype, keylen = unpack('>HHH', fp.read(6))
        keyvalue = fp.read(keylen)
        return CCacheKeyblock(keytype, etype, keyvalue)

    @classmethod
    def write_keyblock(cls, fp, k):
        fp.write(pack('>HHH', k.keytype, k.etype, len(k.keyvalue)))
        fp.write(k.keyvalue)

    @classmethod
    def read_times(cls, fp):
```

```
        authtime, starttime, endtime, renew_till = unpack('>IIII', fp.read(16))
        return CCacheTimes(authtime, starttime, endtime, renew_till)

    @classmethod
    def write_times(cls, fp, t):
        fp.write(pack('>IIII', t.authtime, t.starttime, t.endtime, t.renew_till))

    @classmethod
    def read_address(cls, fp):
        addrtype = unpack('>H', fp.read(2))[0]
        addrdata = cls.read_string(fp)
        return CCacheAddress(addrtype, addrdata)

    @classmethod
    def write_address(cls, fp, a):
        fp.write(pack('>H', a.addrtype))
        cls.write_string(fp, a.addrdata)

    @classmethod
    def read_credential(cls, fp):
        client = cls.read_principal(fp)
        server = cls.read_principal(fp)
        key = cls.read_keyblock(fp)
        time = cls.read_times(fp)
        is_skey, tktflags, num_address = unpack('>BII', fp.read(9))
        addrs = [cls.read_address(fp) for i in range(num_address)]
        num_authdata = unpack('>I', fp.read(4))[0]
        authdata = [cls.read_authdata(fp) for i in range(num_authdata)]
        ticket = cls.read_string(fp)
        second_ticket = cls.read_string(fp)
        return CCacheCredential(client, server, key, time, is_skey, tktflags,
                                addrs, authdata, ticket, second_ticket)

    @classmethod
    def write_credential(cls, fp, c):
        cls.write_principal(fp, c.client)
        cls.write_principal(fp, c.server)
        cls.write_keyblock(fp, c.key)
        cls.write_times(fp, c.time)
        fp.write(pack('>BII', c.is_skey, c.tktflags, len(c.addrs)))
        for addr in c.addrs:
            cls.write_address(fp, addr)
        fp.write(pack('>I', len(c.authdata)))
        for authdata in c.authdata:
            cls.write_authdata(fp, authdata)
        cls.write_string(fp, c.ticket)
        cls.write_string(fp, c.second_ticket)

def get_tgt_cred(ccache):
    for credential in ccache.credentials:
        if credential.server.components[0] == 'krbtgt':
```

```
            return credential
    raise ValueError('No TGT in CCache!')

def kdc_rep2ccache(kdc_rep, kdc_rep_enc):
    return CCacheCredential(
        client=CCachePrincipal(
            name_type=int(kdc_rep['cname']['name-type']),
            realm=str(kdc_rep['crealm']),
            components=[str(c) for c in kdc_rep['cname']['name-string']]),
        server=CCachePrincipal(
            name_type=int(kdc_rep_enc['sname']['name-type']),
            realm=str(kdc_rep_enc['srealm']),
            components=[str(c) for c in kdc_rep_enc['sname']['name-string']]),
        key=CCacheKeyblock(
            keytype=int(kdc_rep_enc['key']['keytype']),
            etype=0,
            keyvalue=str(kdc_rep_enc['key']['keyvalue'])),
        time=CCacheTimes(
            authtime=gt2epoch(str(kdc_rep_enc['authtime'])),
            starttime=gt2epoch(str(kdc_rep_enc['starttime'])),
            endtime=gt2epoch(str(kdc_rep_enc['endtime'])),
            renew_till=gt2epoch(str(kdc_rep_enc['renew-till']))),
        is_skey=0,
        tktflags=bitstring2int(kdc_rep_enc['flags']),
        addrs=[],
        authdata=[],
        ticket=encode(kdc_rep['ticket'].clone(tagSet=Ticket.tagSet,
cloneValueFlag=True)),
        second_ticket='')
```