ДОДАТОК А

Перелік посилань відповідно до наукових досліджень кафедри

2. Kachko O., Bilous N, Semerkov V. Research on methods for secure web applications development // Information Technologies in Innovation Business (ITIB). Proceedings of ITIB, Kharkiv, Ukraine. – 7-9 October 2015. С. 24-27.

6. Єрохін А., Пашкевич Є. Исследование методов и средств мониторинга серверов // Системи обробки інформації. 2011. №4. С. 188-192.

13. Maksym Bekuzarov, Mariya Shirokopetleva, Oleksandr Samantsov, Oksana Mazurova Neural Network Architecture Editor With Code Generation // Problem of Infocommunications. Science and Technolpgy (PIC S&T′2020), Kharkiv, Ukraine. – 6-9 October 2020 (Scopus). С. 35-39.

14. Мазурова О., Широкопетлева М. Поддержка Проектирования Баз Данных // Інформаційні системи та технології: матеріали міжнар. наук.-техн. конф, Харків, Україна. – 10-15 вересня 2018. С. 319-322.

23. Andrey Arsenov, Igor Ruban, Kyrylo Smelyakov, Anastasiya Chupryna . Evolution of Convolutional Neural Network Architecture in Image Classification Problems // Selected Papers of the XVIII International Scientific and Practical Conference on IT and Security (ITS). Kiyv, Ukraine. 2019. С. 9-10.

ДОДАТОК Б
Слайди презентації

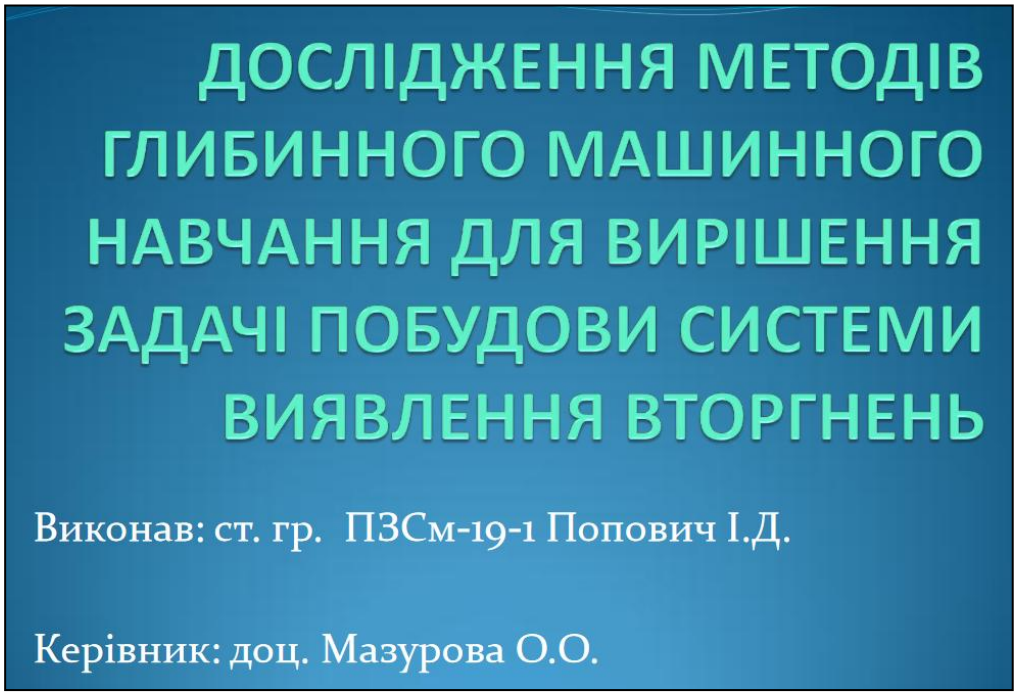

Рисунок Б.1 – Вступний слайд

Рисунок Б.2 – Аналіз проблемної області

Рисунок Б.3 – Аналоги

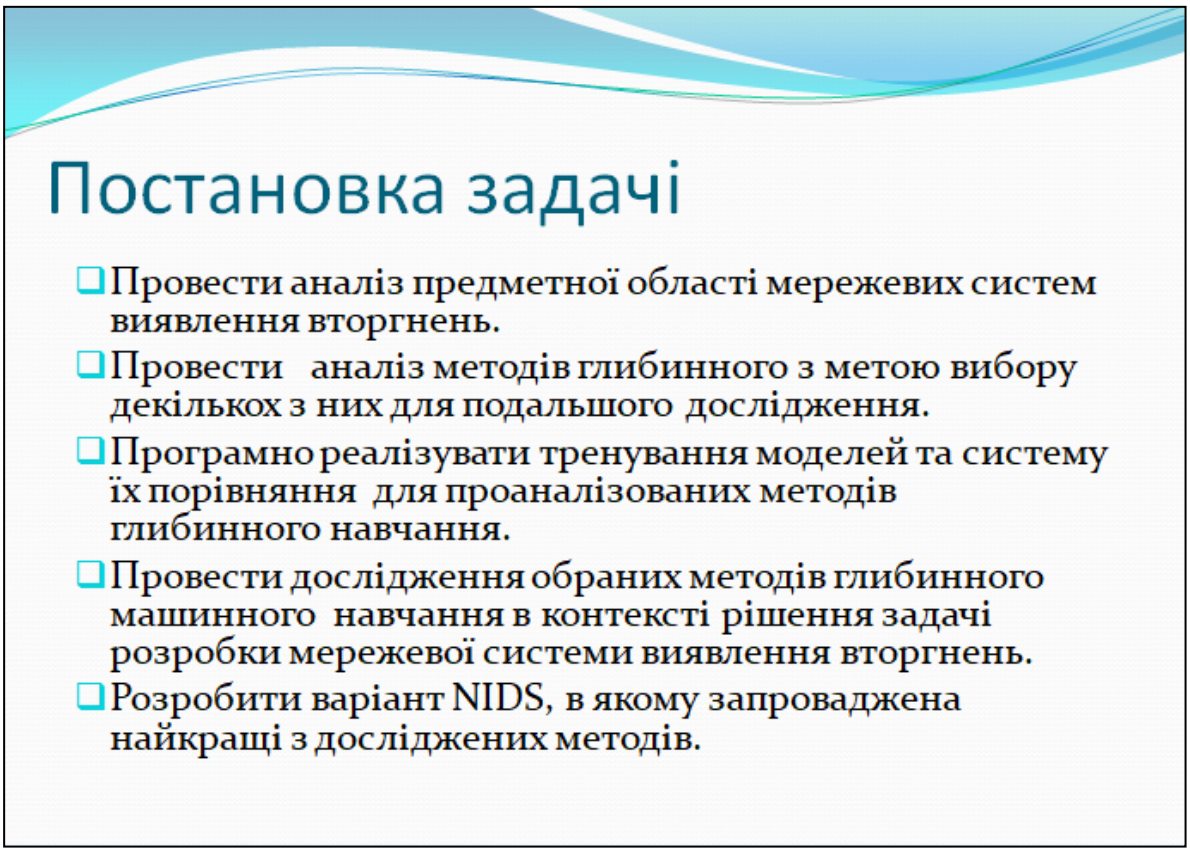

Рисунок Б.4 – Постановка задачі

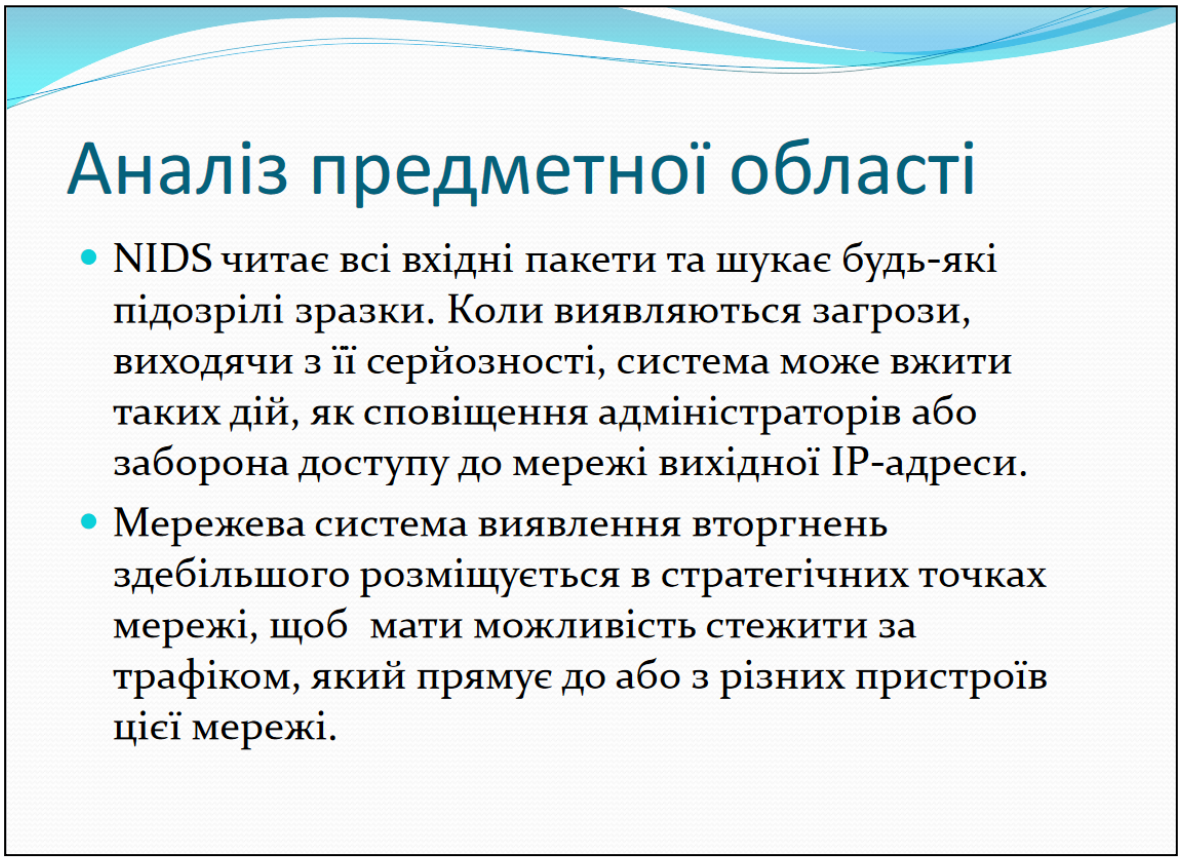

Рисунок Б.5 – Аналіз проблемної області

Рисунок Б.6 – Методи машинного навчання

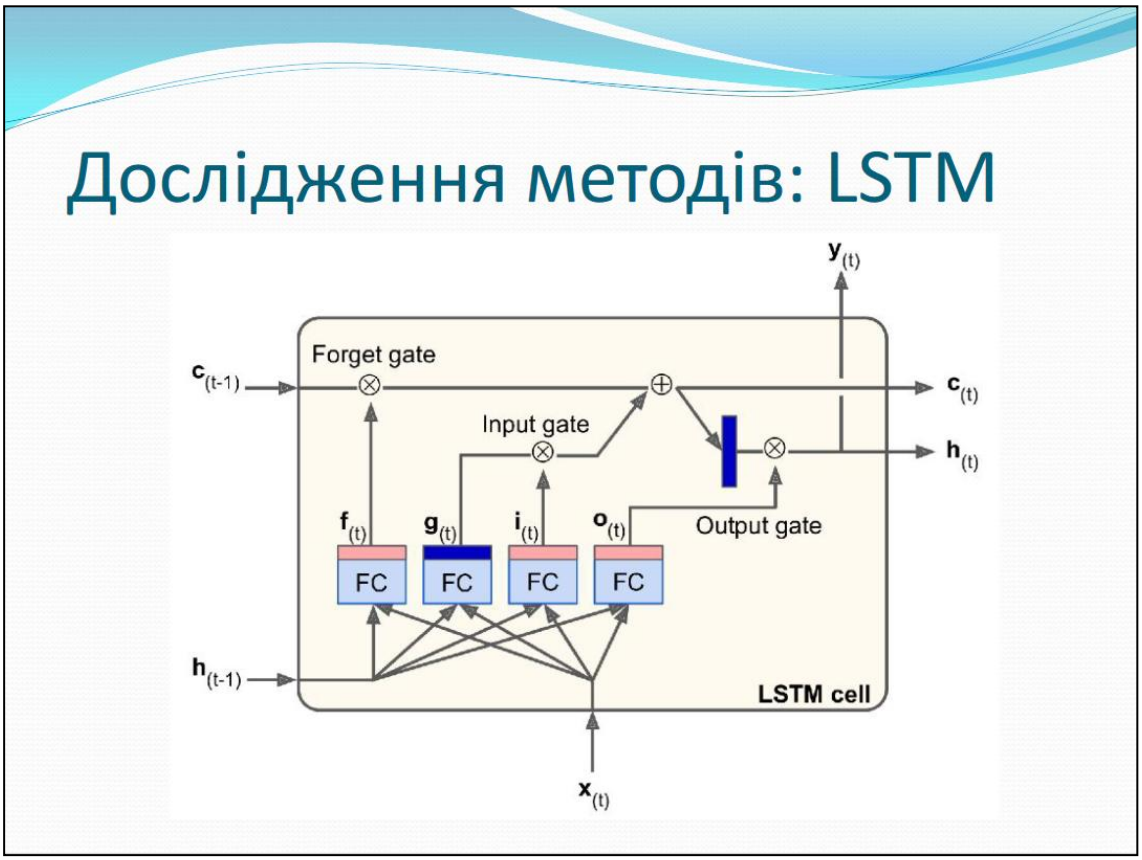

Рисунок Б.7 – Дослідження методів: LSTM

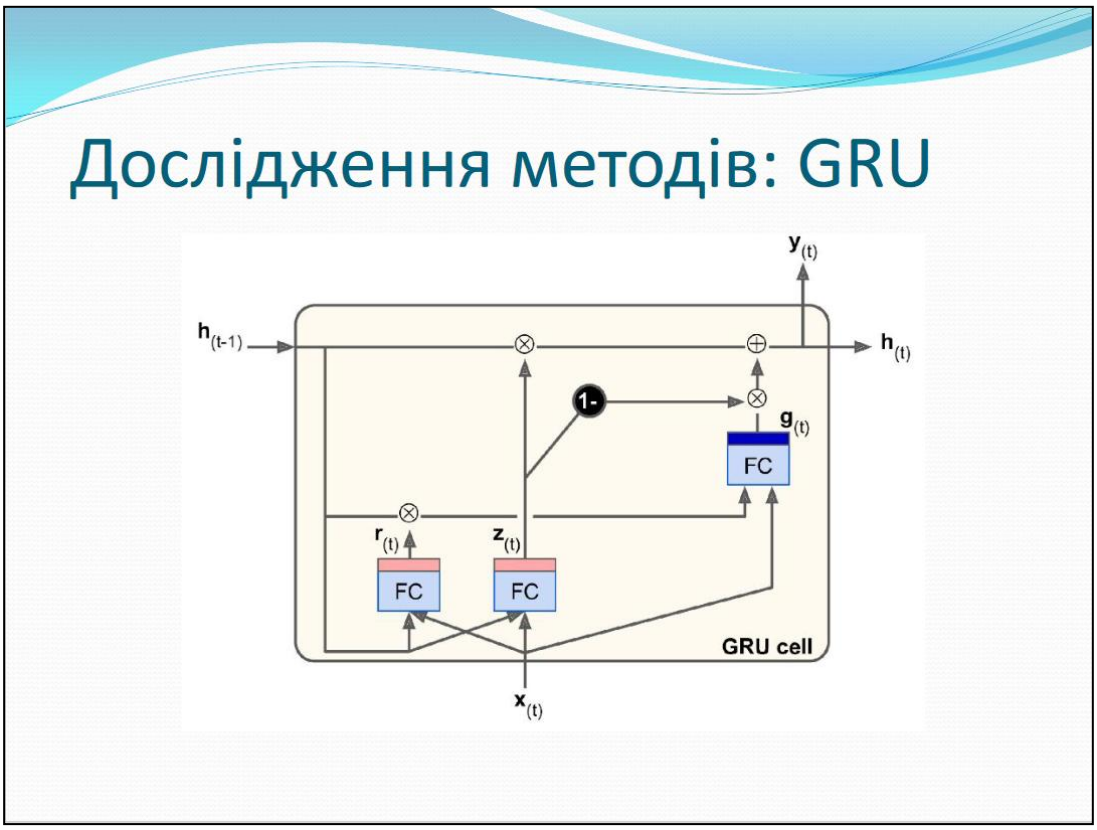

Рисунок Б.8 – Дослідження методів: GRU

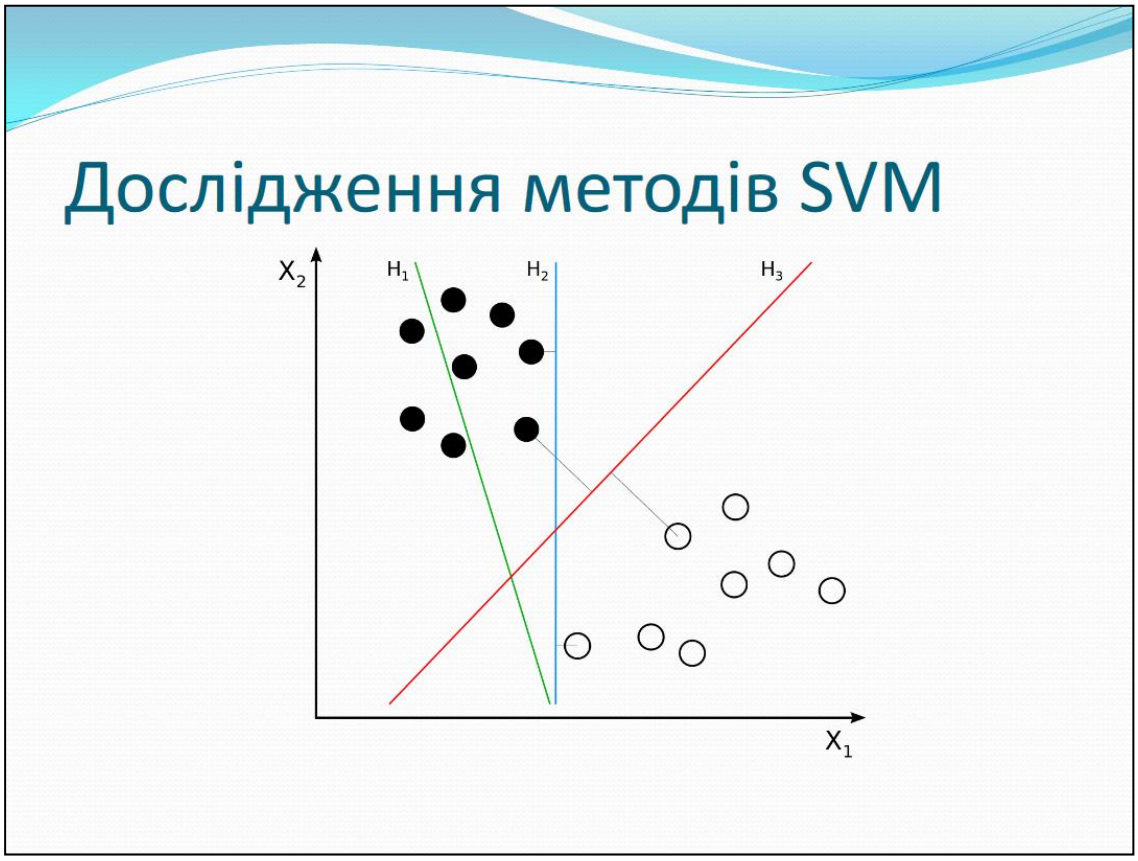

Рисунок Б.9 – Дослідження методів: SVM

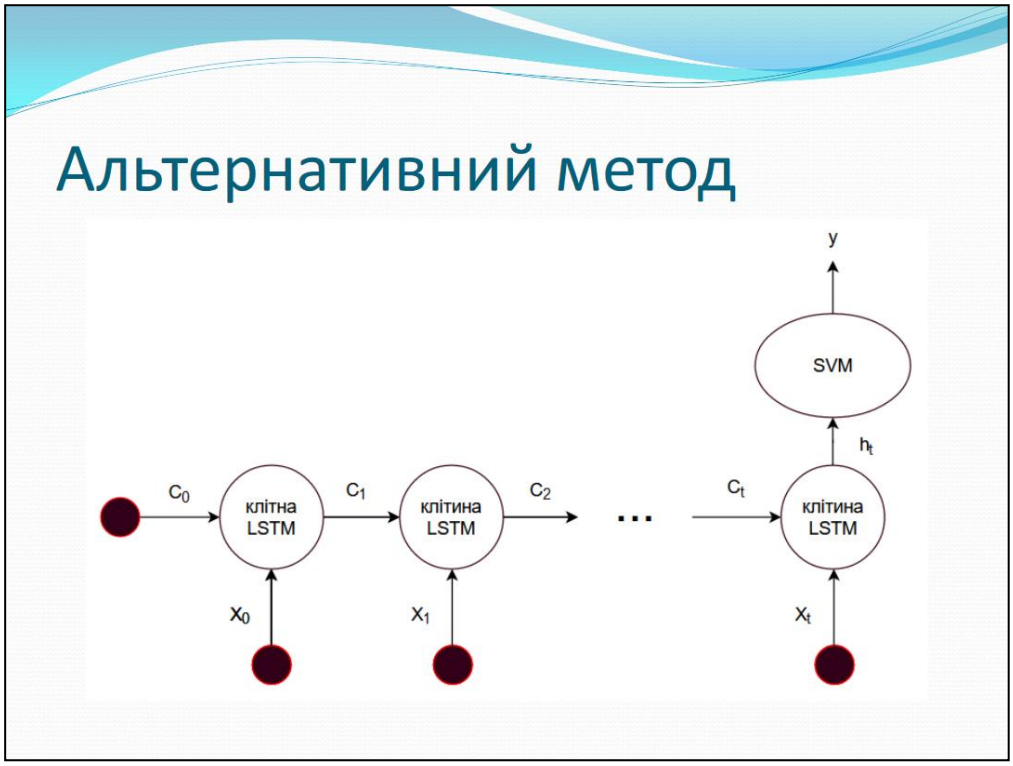

Рисунок Б.10 – Альтернативний метод

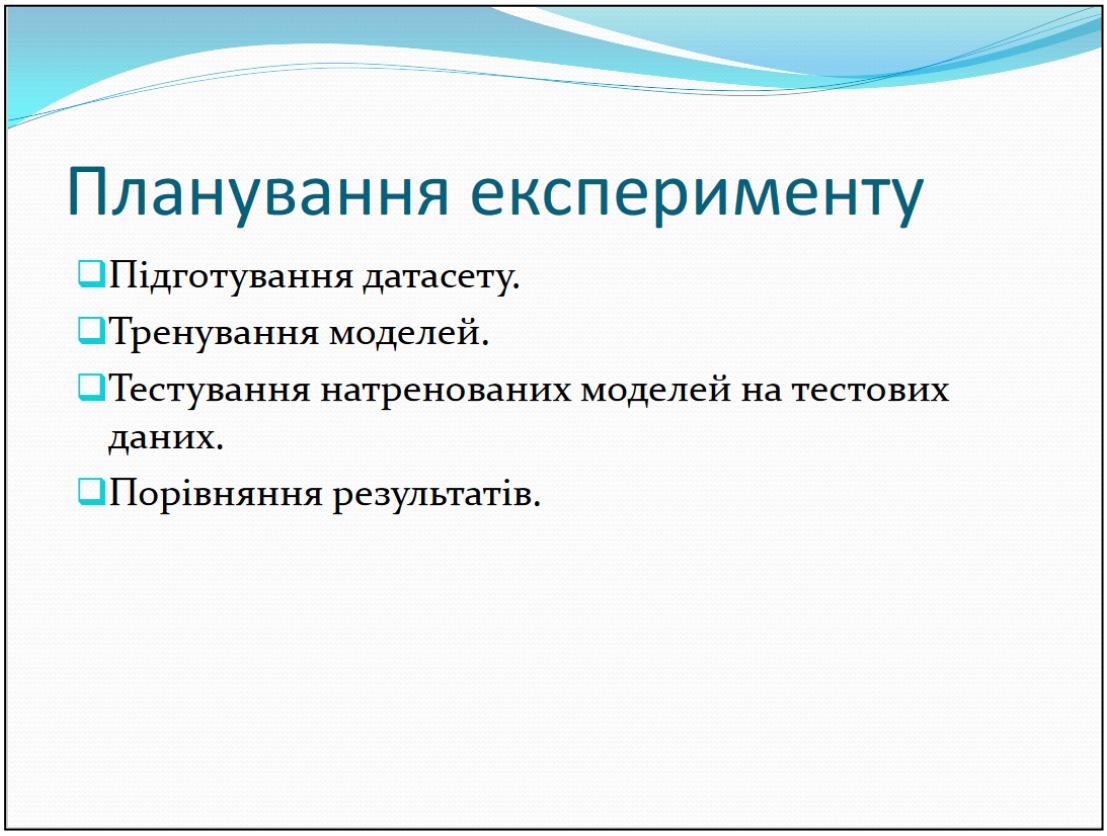

Рисунок Б.11 – Планування експерименту

Рисунок Б.12 – Підготування датасету

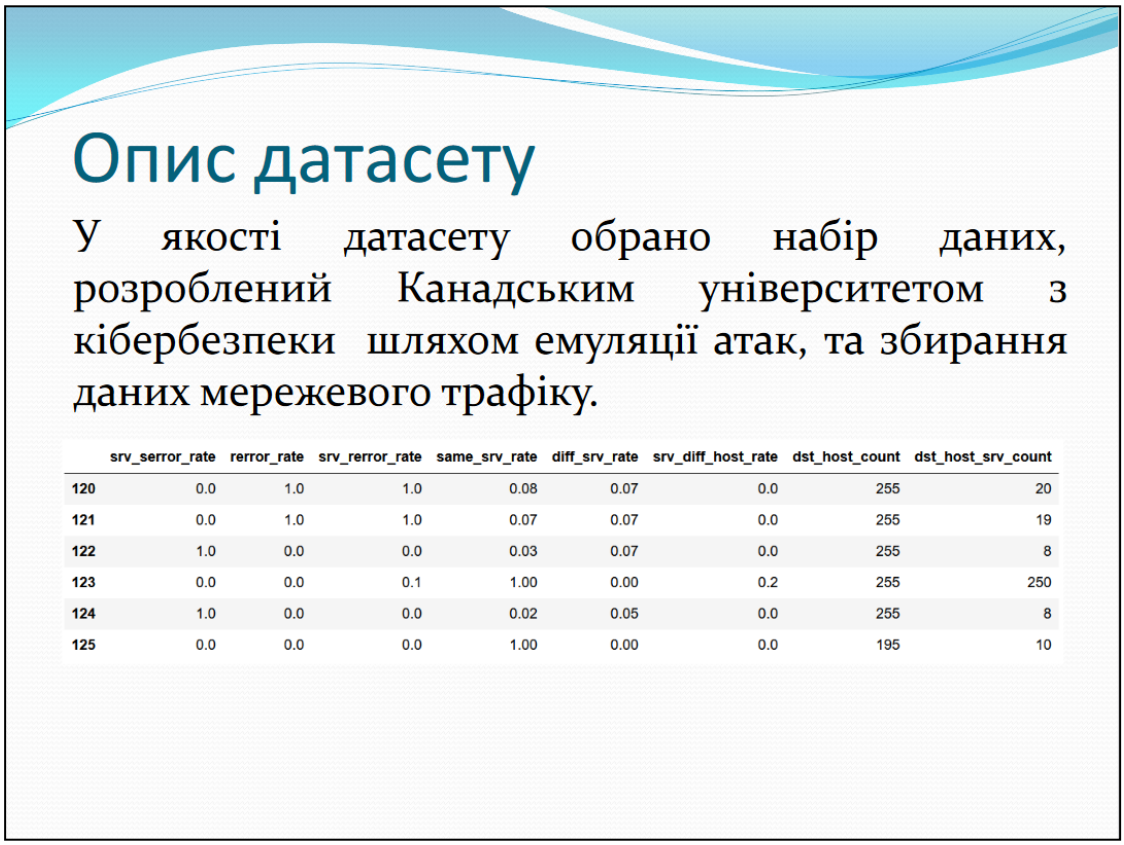

Рисунок Б.13 – Опис датасету

Рисунок Б.14 – Гіперпараметри моделей

Рисунок Б.15 – Середа для тренування

Рисунок Б.16 – Результати експерименту

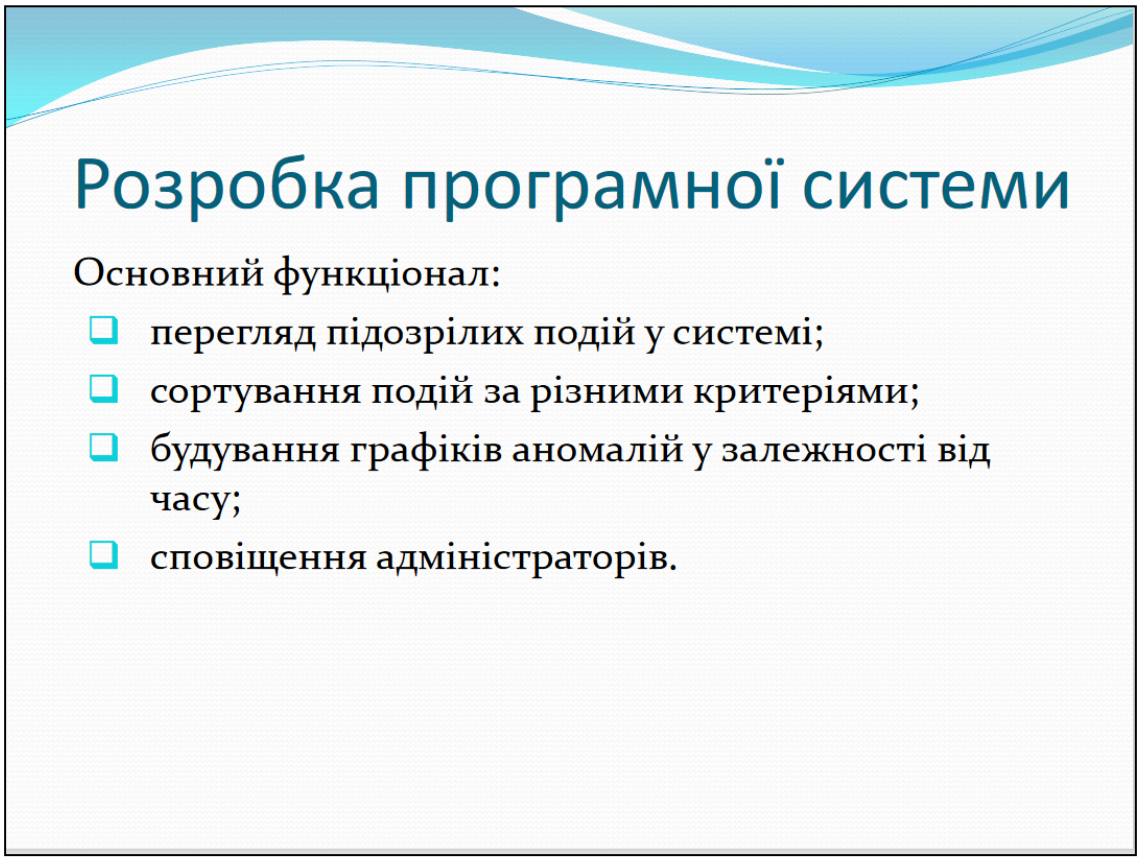

Рисунок Б.17 – Розробка програмної системи

Рисунок Б.18 – Схема бази даних

Рисунок Б.19 – Діаграма розгортання

Рисунок Б.20 – Діаграма компонентів

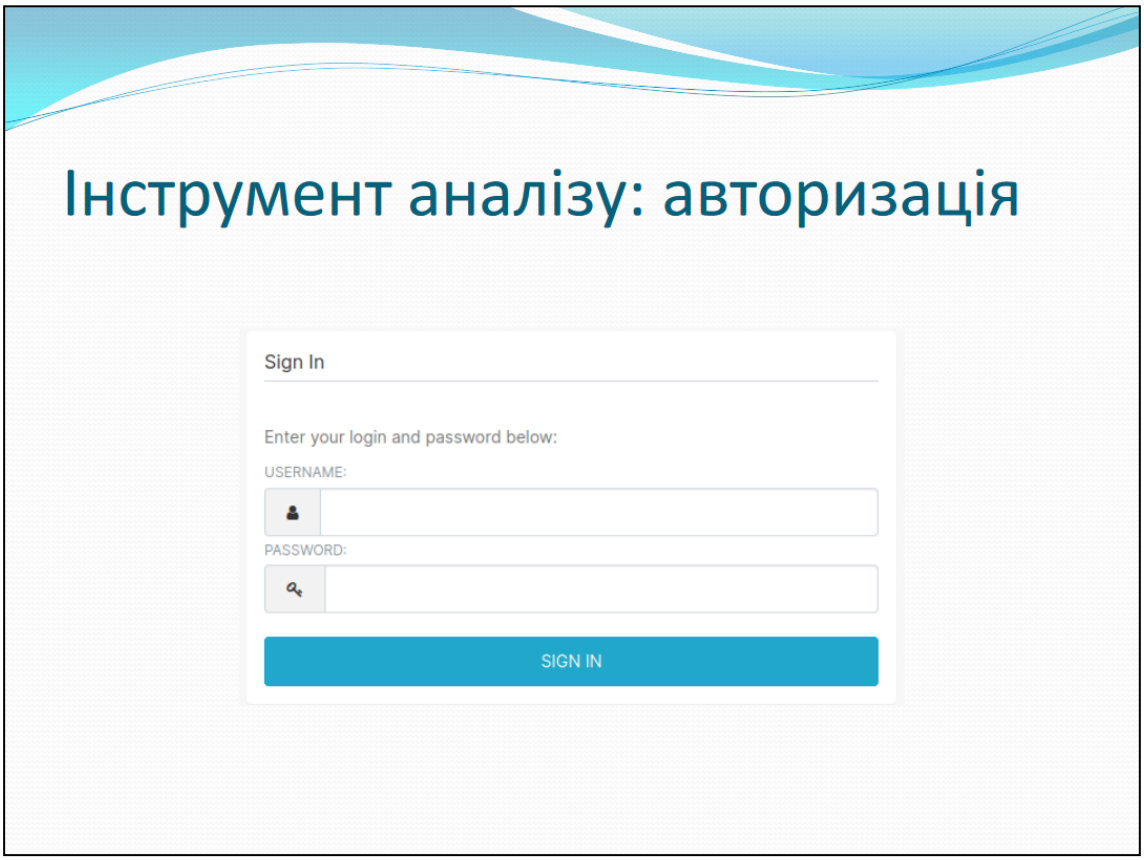

Рисунок Б.21 – Інструмент аналізу: авторизація

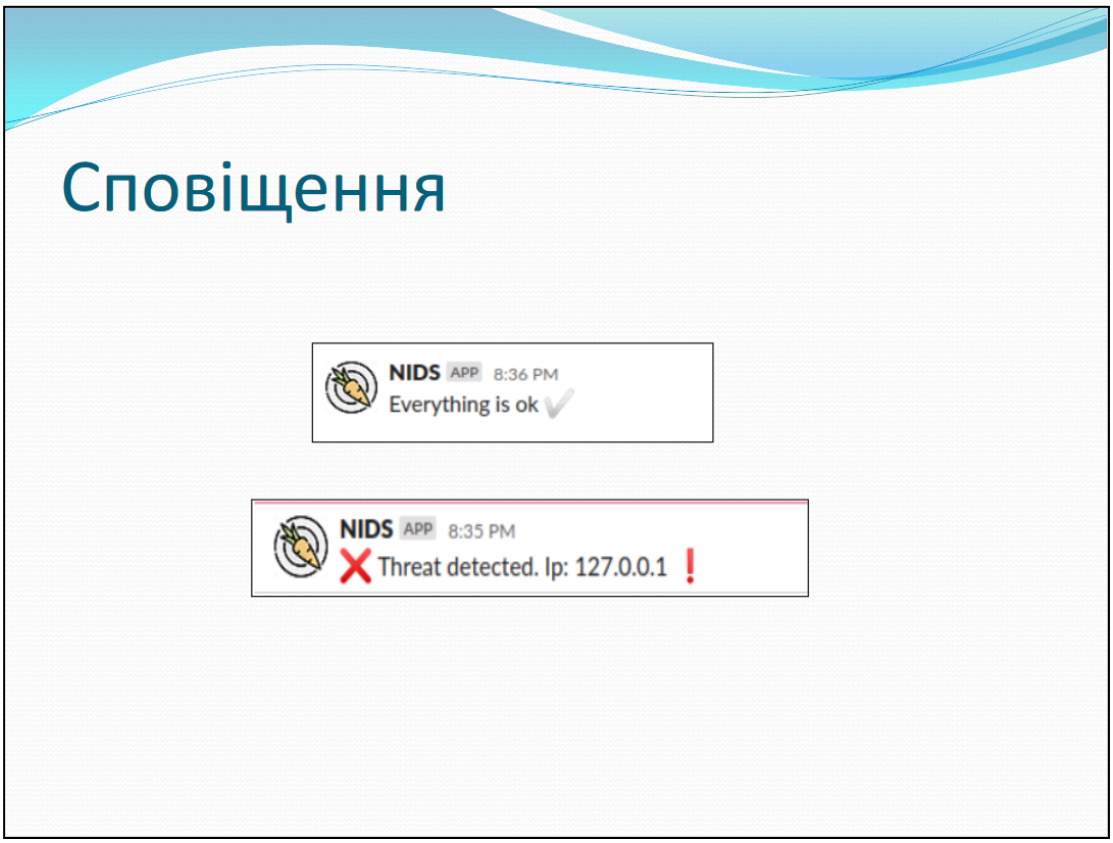

Рисунок Б.22 – Інструмент аналізу: сповіщення

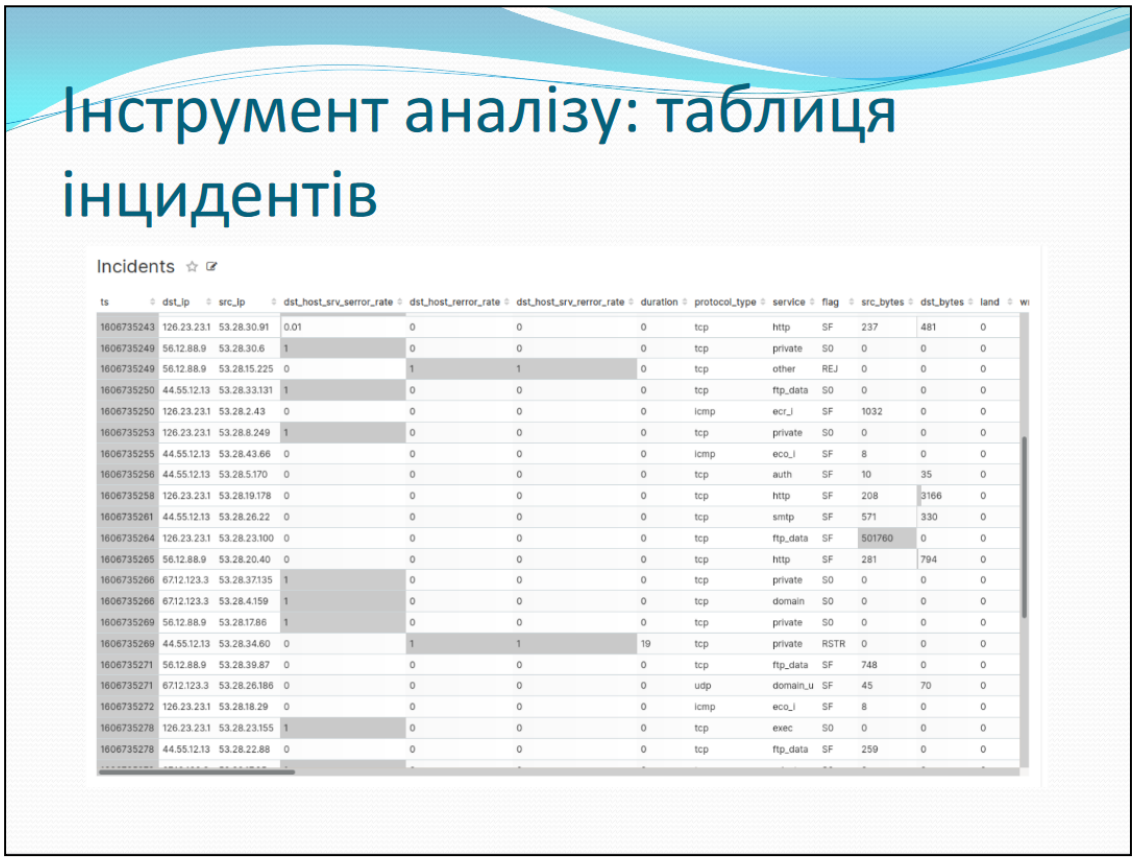

Рисунок Б.23 – Інструмент аналізу: таблиця інцидентів

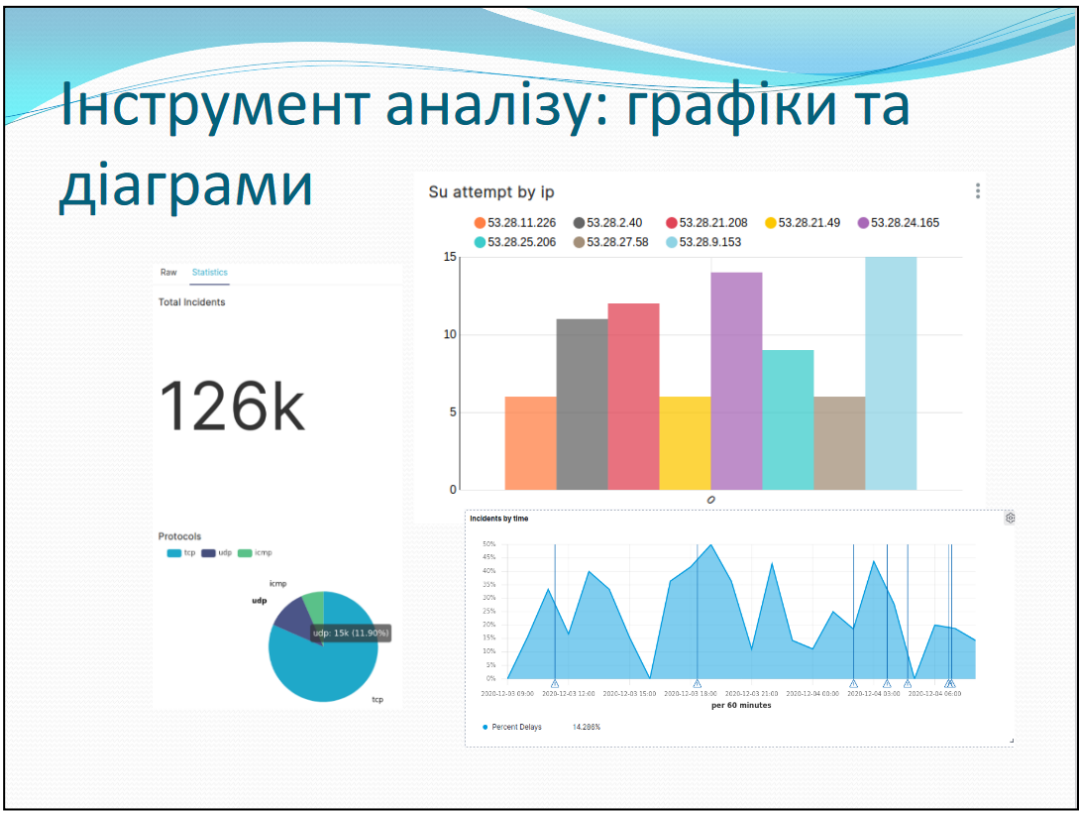

Рисунок Б.24 – Інструмент аналізу: графіки та діаграми

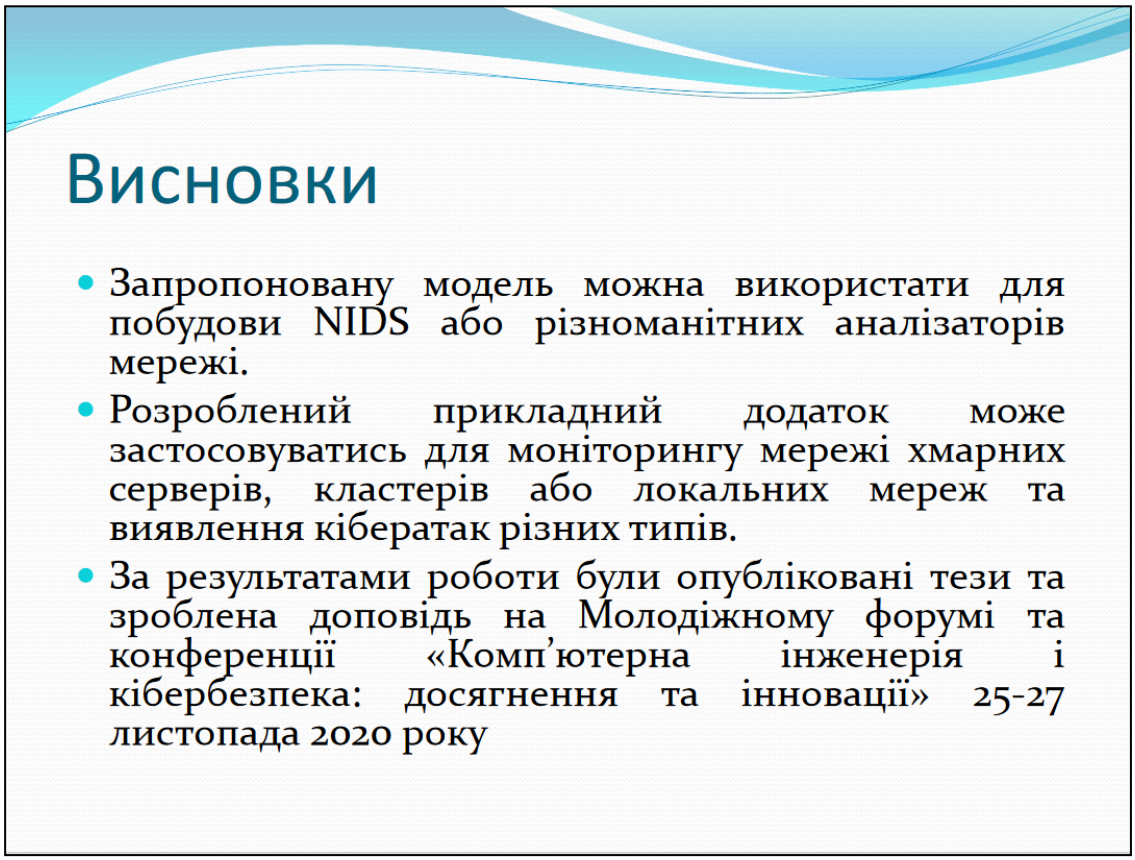

Рисунок Б.25 – Висновки

ДОДАТОК В

Тези доповіді на молодіжний форум

СИСТЕМА ВИЯВЛЕННЯ ВТОРГНЕНЬ НА ОСНОВІ МЕТОДІВ ГЛИБИННОГО НАВЧАННЯ

Попович І.Д.
Науковий керівник – к.т.н., доцент Мазурова О.О.
Харківський національний університет радіоелектроніки
(61166, Харків, пр. Науки, 14, каф. Програмної інженерії, тел. (057) 702-14-46)
e-mail: ivan.popovych@nure.ua; телефон (066) 295-37-10

Security of cloud based servers is one of the most important aspects of modern web-applications. Intrusion detection systems are effective instruments to detect incoming threats to be able to react in a proper way. Rapid artificial intelligence technology growing provides a great opportunity to make Intrusion detection systems more efficient. In order to analyze the effectiveness of different deep learning approaches, created a web tool that used pretrained models to solve intrusion detection system task.

Безпека хмарних додатків є одним з найважливіших завдань, яке розв'язується у ході проектування сучасних веб-систем. В найкращому варіанті необхідно розробити систему, яка повністю захищена від зловмисників. Але в реальному випадку це неможливо. Тому, щоб своєчасно реагувати, потрібно якнайшвидше визначити загрозу.

Система виявлення вторгнень (англ. Intrusion detection system — IDS) — це тип програмного забезпечення, призначеного для автоматичного сповіщення адміністраторів, коли хтось чи щось намагається порушити інформаційну систему через зловмисні дії або через порушення правил безпеки [1].

IDS може містити один або обидва типи виявлення вторгнень: на основі визначення підписів та на основі визначення аномалій [2]. IDS на основі підписів відстежує мережевий трафік на предмет підозрілих зразків у пакетах даних, підписів відомих вторгнень у мережу, щоб виявити та усунути напади та компроміси [2]. Це досягається за допомогою використання бази даних відомих типів вторгнень та шаблонів даних, що дозволяє IDS на основі підписів швидко ідентифікувати вторгнення та розпочати відповідний хід дій.

В свою чергу IDS на основі аномалії використовує властивості системи у звичайному стані, щоб відстежувати, чи відбувається незвична чи підозріла діяльність [2]. Цей метод вимагає підготовки, оскільки для базової підготовки моделі потрібно, щоб IDS дізнався про ваші схеми використання, і це робить його орієнтичним, евристичним підходом до виявлення вторгнень. Перевага IDS на основі аномалії полягає в тому, що він є більш гнучким та потужним, ніж IDS на основі підписів, для якої необхідна існуюча база даних зловмисного трафіку.

Отже, була поставлена задача дослідити підходи, які можна використати для створення системи виявлення вторгнень на основі аномалії, побудувати модель, яка базується на обраних підходах та програмно реалізувати її в складі прикладного додатку, призначеного для виявлення аномальної активності в мережі у реальному часі.

Дослідження проводилися в області штучного інтелекту. Досягнення у цьому напрямку активно застосовуються у багатьох галузях, у тому числі і в галузі безпеки програмного забезпечення. Для розв'язання задачі визначення аномалій існує багато методів класичного машинного навчання, такі як нейронні мережі (ANN), Support vector machines (SVM), Naïve-Bayesian (NB), Random Forests (RF), Self-Organized Maps (SOM) . Наразі активно використовують більш ефективний підхід глибинного навчання. Зокрема, звичайні рекурентні нейронні мережі (RNN), а також більш специфічні Long short-term memory(LSTM) та Gated recurrent unit (GRU) [3].

Головна перевага архітектури LSTM над звичайними рекурентними нейронними мережами в тому, що вона може обробляти довготривалі залежності у даних. Це стало можливим завдяки додаванню до одиниці мережі додаткових шарів, які регулюють потік інформації до мережевої клітини.

GRU створений для вирішення тієї ж проблеми, що LSTM, але її одиниця мережі має на один шар менше. Крім того, GRU має можливість повторювати останні вихідні дані при переході до наступної ітерації мережі. Це робить її більш ефективною, ніж LSTM, в деяких випадках, зокрема, на датасетах малого розміру.

У даній роботі проаналізовано методи Long short-term memory та Gated recurrent unit у контексті задачі навчання для побудови IDS на основі аномалій та реалізовано демонстраційну систему виявлення вторгнень, яка базується на натренованих моделях, побудованих за допомогою обраних методів:

- перегляд підозрілих подій у системі;
- відображення статусу підозрілих на аномалію подій;
- сортування подій за різними критеріями;
- будування графіків аномалій у залежності від часу.

У якості технології для тренування моделей було використано мову Python та фреймворк для глибинного навчання TensorFlow. Для написання прикладного додатку було використано Python-фреймворк aiohttp на базі asyncio для написання API для натренованих моделей та нереляційну базу даних MongoDB у якості базового сховища даних. Також було використано Vue.js для побудови веб-клієнту системи моніторингу.

Запропоновані моделі може бути використано для побудови IDS або різноманітних аналізаторів мережі. Розроблений прикладний додаток може бути використано для моніторінгу мережі різних хмарних серверів, кластерів, або локальних мереж та виявлення різноманітних кібератак.

Список використаних джерел: 1. Michael Sikorski, Andrew Honig. Practical Malware Analysis. – No Strach Press, 2018. – 802 с.: ил; 2. Stefan Axelsson. Intrusion Detection Systems: A Survey and Taxonomy. – Department of Computer Engineering Chalmers University of Technology G¨oteborg, Sweden, 2015. – 27 с.: ил.; 3. Raghavendra Chalapathy, Sanjay Chawla. Deep Learning for Anomaly Detection: A Survey. – University of Sydney, 2019. – 48 с.: ил.

ДОДАТОК Г

Тези на конференцію «Комп’ютерна інженерія: досягнення та інновації»

УДК 004.02 :004.056.5

І.Д. Попович
студент факультету комп’ютерних наук,
Харківський національний університет
радіоелектроніки

ДОСЛІДЖЕННЯ МЕТОДІВ ГЛИБИННОГО МАШИННОГО НАВЧАННЯ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ ПОБУДОВИ СИСТЕМИ ВИЯВЛЕННЯ ВТОРГНЕНЬ

Забезпечення інформаційної безпеки є одним з найважливіших завдань, яке розв’язується у ході проектування сучасних систем [1]. В реальних умовах практично неможливо мати систему, яка повністю захищена від зловмисних дій, але вона повинна хоча б своєчасно реагувати, а для цього якнайшвидше визначати загрозу.

Мережева система виявлення вторгнень (англ. Network Intrusion detection system – NIDS) – це програмне забезпечення, призначене для аналізу мережевого трафіку та виявлення зловмисних інцидентів у ньому [2]. NIDS поділяють на системи, які базуються на основі підписів та на основі аномалій. Перші відстежують мережевий трафік на предмет підозрілих зразків у пакетах даних за допомогою бази даних підписів відомих вторгнень у мережу [2]. NIDS на основі аномалії використовує системи у звичайному стані, щоб відстежувати, чи відбувається незвична або підозріла діяльність.

Формально NIDS на основі аномалій виконує функцію класифікації трафіку на шкідливий і звичайний [2]. Зараз завдання класифікації досить ефективно вирішують за допомогою методів машинного навчання. Для розв’язання задачі виявлення аномалій існує багато методів класичного машинного навчання, такі як нейронні мережі, байєсівські мережі, метод опорних векторів тощо. Наразі, активно використовують більш ефективні підходи глибинного навчання, зокрема нейронні мережі.

Отже, було поставлено задачу дослідити методи машинного навчання, зокрема різні типи архітектур нейронних мереж, а саме, натренувати модель, визначивши їх оптимальні параметри на основі даних для навчання, та порівняти їх ефективність. Найбільш ефективні модел використати для створення системи виявлення вторгнень.

Для проведення дослідження були обрані архітектури наступних мереж: згорткова нейронна мережа (англ. Convolutional neural network – CNN), рекурентна нейронна мережа (англ. Recurrent neural network – RNN), а саме її різновиди: довга короткочасна пам’ять (англ. Long short-term memory – LSTM) та керуючий рекурентний блок (англ. Gated recurrent unit – GRU) [3].

Особливість методу CNN полягає у тому, що за основу береться математична операція, яка називається згорткою – спеціалізований вид лінійних операцій [3]. CNN зазвичай використовують для вирішення завдань класифікації зображень, але в випадку спеціальної обробки даних перед тренуванням та при використанні особливої оптимізації ця архітектура може бути використана для виявлення аномалій.

Нейронні мережі, які засновані на архітектурі LSTM активно використовуються для аналізу природної мови [3]. Клітина LSTM запам’ятовує значення через довільні інтервали часу, тому такі мережі також добре підходять для класифікації, обробки та виявлення аномалій на основі даних часових рядів, оскільки важливі події можуть виникати через невизначені проміжки. Тобто LSTM може досить ефективно вирішувати завдання визначення зловмисного трафіку у мережі.

Архітектура GRU фактично є більш простою версією LSTM, але її одиниця мережі має на один шар менше [3]. Це дозволяє отримати більш високі результати у випадку, якщо кількість інформації є порівняно невеликою, а у випадку аналізу мережевого трафіку зазвичай доступні дані за період до одного тижня.

Також було вирішено дослідити доцільність використання комбінації підходів LSTM та методу SVM, який був використаний у якості класифікатору.

У якості датасету обрано набір даних, розроблений Канадським університетом з кібербезпеки [4] шляхом емуляції найбільш поширених атак, та збирання даних мережевого трафіку впродовж п’яти днів у стандартному форматі pcap. Для подальшого його використання для тренування моделей, датасет було перетворено до формату XML. З XML файлу вилучено параметри мережі та сконвертовано до масиву, а також застосовано операцію one hot encoding [3] для отримання бінарних значень параметру (присутній або ні). Отримані дані було розділено на частини для тренування та для тесту. У ході роботи кожна модель натренована за допомогою методу гралієнтного спуску на першій частині даних, тобто були отримані оптимальні вагові параметрів нейронної мережі, а після цього протестовано на другій частині даних. Ефективність моделей визначалася за допомогою наступних критеріїв:

– точність – процент правильно визначених загроз;
– вірні позитивні відповіді – процент правильно визначених аномалій;
– невірні позитивні відповіді – процент неправильно визначених аномалій.

В ході дослідження отримано результати, які наведені у таблиці 1.

Таблиця 1 – **Результати дослідження**

	Точність (%)	Вірні позитивні відповіді (%)	Невірні позитивні відповіді (%)
CNN	95,4	93,6	3,3
LSTM	96,8	94,9	2,5
GRU	96,4	95,2	3,1
LSTM+SVM	96,1	94,7	2,7

Отримані результати дозволяють зробити висновки, що архітектура GRU та LSTM є ефективнішими за CNN. При цьому LSTM показує дещо більш високу точність, але GRU має менше невірних позитивних відповідей. Комбінація методів LSTM + SVM значно не покращила ефективність мережі.

На базі отриманих результатів для реалізації було обрано модель на архітектурі GRU, за допомогою якої реалізовано демонстраційну систему виявлення вторгнень. Розроблена система забезпечує наступний функціонал: перегляд підозрілих подій у системі, відображення статусу підозрілих на аномалію подій, сортування подій за різними критеріями, будування графіків аномалій у залежності від часу, тощо.

Запропоновану модель можна використати для побудови NIDS або різноманітних аналізаторів мережі. Розроблений прикладний додаток може застосовуватись для моніторингу мережі хмарних серверів, кластерів або локальних мереж та виявлення кібератак різних типів.

Список використаних джерел:

1. Бобало Ю.Я. Інформаційна безпека / Ю.Я. Бобало, І.В. Горбатий, М.Д. Кісельчинк, А.П. Бомарка та ін. – Львів: Львівська політехніка, 2019. – 580 с.
2. Newman R. Computer Security; Protecting Digital Resources. – New York: Jones & Bartlett Learning, 2009. – 336 p.
3. Raghavendra Chalapathy, Sanjay Chawla. Deep Learning for Anomaly Detection: A Survey. – University of Sydney, 2019. – 48 с.: кл.
4. Canadian Institute for Cybersecurity datasets URL: https://www.unb.ca/cic/datasets/ (дата звернення: 29.10.2020).

Науковий керівник – кандидат технічних наук, доцент Мазурова О.О., доцент кафедри програмної інженерії Харківського національного університету радіоелектроніки.

ДОДАТОК Д
Лістінг коду

```
import glob
import json
import logging
import os
import time

import request


logger = logging.getLogger(__name__)

ENDPOINT = 'http://127.0.0.1:10000'
BASE_DIR = os.getnev('BASE_DIR')
POS_FILEPATH = os.join(BASE_DIR, 'pos.json')


def send_pcap(pcap_filepath):
    with open(pcap_filepath) as pcap_file:
        request.post(ENDPOINT, data=pcap_file)


def get_last_ts():
    with open(POS_FILEPATH) as pos_file:
        data = json.load(pos_file)
        return data['ts']
```

```
def save_last_ts():
    ts = time.time()
    with open(POS_FILEPATH, 'w') as pos_file:
        json.dump({'ts': ts}, pos_file)



def send_files():
    ts = get_last_ts()
    pcap_files = os.path.join(f'{BASE_DIR}', '*.cap')
    for filepath in glob.glob(pcap_files):
        if os.path.getmtime(filepath) > ts:
            send_pcap(filepath)
    save_last_ts()



def main():
    send_files()



if __name__ == '__main__':
    main()
```

```
const propTypes = {
  depth: PropTypes.number.isRequired,
  editMode: PropTypes.bool.isRequired,
  gridComponent: componentShape.isRequired,
  handleComponentDrop: PropTypes.func.isRequired,
```

```
  isComponentVisible: PropTypes.bool.isRequired,
  resizeComponent: PropTypes.func.isRequired,
  setDirectPathToChild: PropTypes.func.isRequired,
  width: PropTypes.number.isRequired,
};

const defaultProps = {};

class DashboardComponent extends Component {
  constructor(props) {
    super(props);
    this.state = {
      Resizing: false,
      rowGuideTop: null,
    };
  }

  setGridRef(ref) {
    this.grid = ref;
  }

  handleResizeStart({ ref, direction }) {
    let rowGuideTop = null;
    if (direction === 'bottom' || direction === 'bottomRight') {
      rowGuideTop = this.getRowGuidePosition(ref);
    }

    this.setState(() => ({
      Resizing: true,
      rowGuideTop,
```

```
  }));
}

handleResize({ ref, direction }) {
  if (direction === 'bottom' || direction === 'bottomRight') {
    this.setState(() => ({ rowGuideTop: this.getRowGuidePosition(ref) }));
  }
}

handleResizeStop({ id, widthMultiple: width, heightMultiple: height }) {
  this.props.resizeComponent({ id, width, height });

  this.setState(() => ({
    Resizing: false,
    rowGuideTop: null,
  }));
}

handleTopDropTargetDrop(dropResult) {
  if (dropResult) {
    this.props.handleComponentDrop({
      ...dropResult,
      destination: {
        ...dropResult.destination,
        index: 0,
      },
    });
  }
}
```

```
handleChangeTab({ pathToTabIndex }) {
  this.props.setDirectPathToChild(pathToTabIndex);
}

render() {
  const {
    gridComponent,
    handleComponentDrop,
    depth,
    editMode,
    width,
    isComponentVisible,
  } = this.props;

  const columnPlusGutterWidth =
    (width + GRID_GUTTER_SIZE) / GRID_COLUMN_COUNT;

  const columnWidth = columnPlusGutterWidth - GRID_GUTTER_SIZE;
  const { Resizing, rowGuideTop } = this.state;

  return width < 150 ? null : (
    <div className="dashboard-grid" ref={this.setGridRef}>
      <div className="grid-content" data-test="grid-content">
        {}
        {editMode && (
          <DragDroppable
            component={gridComponent}
            depth={depth}
            parentComponent={null}
            index={0}
```

```
    orientation="column"
    onDrop={this.handleTopDropTargetDrop}
    className="empty-droptarget"
    editMode
  >
    {({ dropIndicatorProps }) =>
      dropIndicatorProps && (
        <div className="drop-indicator drop-indicator--bottom" />
      )
    }
  </DragDroppable>
)}

{gridComponent.children.map((id, index) => (
  <Component
    key={id}
    id={id}
    parentId={gridComponent.id}
    depth={depth + 1}
    index={index}
    availableColumnCount={GRID_COLUMN_COUNT}
    columnWidth={columnWidth}
    isComponentVisible={isComponentVisible}
    onResizeStart={this.handleResizeStart}
    onResize={this.handleResize}
    onResizeStop={this.handleResizeStop}
    onChangeTab={this.handleChangeTab}
  />
))}
```

```
{}
{editMode && gridComponent.children.length > 0 && (
 <DragDroppable
  component={gridComponent}
  depth={depth}
  parentComponent={null}
  index={gridComponent.children.length}
  orientation="column"
  onDrop={handleComponentDrop}
  className="empty-droptarget"
  editMode
 >
  {({ dropIndicatorProps }) =>
   dropIndicatorProps && (
    <div className="drop-indicator drop-indicator--top" />
   )
  }
 </DragDroppable>
)}

{Resizing &&
 Array(GRID_COLUMN_COUNT)
  .fill(null)
  .map((_, i) => (
   <div
    key={`column-grid-${i}`}
    className="column-grid"
    style={{
     left: i * GRID_GUTTER_SIZE + i * columnWidth,
     width: columnWidth,
```

```
              }}
            />
          ))}

        {Resizing && rowGuideTop && (
          <div
            className="row-grid"
            style={{
              top: rowGuideTop,
              width,
            }}
          />
        )}
      </div>
    </div>
  );
 }
}

class DatabaseRestApi(RestAPI):
    datamodel = SQLInterface(Database)

    @expose("/", methods=["POST"])
    def post(self) -> Response:
        if not request.is_json:
            return self.response_400(message="Request is not JSON")
        try:
            item = self.add_model_schema.load(request.json)
        except ValidationError as error:
            return self.response_400(message=error.messages)
```

```
    try:
        new_model = CreateDatabaseCommand(g.user, item).run()
        item["sqlalchemy_uri"] = new_model.sqlalchemy_uri
        return self.response(201, id=new_model.id, result=item)
    except DatabaseCreateFailedError as ex:
        return self.response_422(message=str(ex))

@expose("/<int:pk>", methods=["PUT"])
def put(self, pk: int) -> Response:
    try:
        changed_model = UpdateDatabaseCommand(g.user, pk, item).run()
        item["sqlalchemy_uri"] = changed_model.sqlalchemy_uri
        return self.response(200, id=changed_model.id, result=item)
    except DatabaseUpdateFailedError as ex:
        return self.response_422(message=str(ex))

@expose("/<int:pk>", methods=["DELETE"])
def delete(self, pk: int) -> Response:
    try:
        DeleteDatabaseCommand(g.user, pk).run()
        return self.response(200, message="OK")
    except DatabaseDeleteError as ex:
        return self.response_422(message=str(ex))


@expose("/<int:pk>/table/<table_name>/<schema_name>/", methods=["GET"])
def table_metadata(
    self, database: Database, table_name: str, schema_name: str
) -> Response:
    self.incr_stats("init", self.table_metadata.__name__)
```

```
    try:
        table_info = metadata(database, table_name, schema_name)
    except SQLAlchemyError as ex:
        self.incr_stats("error", self.table_metadata.__name__)
        return self.response_422(error_msg_from_exception(ex))
    self.incr_stats("success", self.table_metadata.__name__)
    return self.response(200, **table_info)

@expose("/<int:pk>/select_star/<table_name>/", methods=["GET"])
def select_star(
    self, database: Database, table_name: str, schema_name: Optional[str] = None
) -> Response:
    self.incr_stats("init", self.select_star.__name__)
    try:
        result = database.select_star(
            table_name, schema_name, latest_partition=True, show_cols=True
        )
    except NoSuchTableError:
        self.incr_stats("error", self.select_star.__name__)
        return self.response(404, message="Table not found on the database")
    self.incr_stats("success", self.select_star.__name__)
    return self.response(200, result=result)

@expose("/test_connection", methods=["POST"])
def test_connection(
    self,
) -> Response:
    if not request.is_json:
        return self.response_400(message="Request is not JSON")
    try:
```

```
            item = DatabaseConnection().load(request.json)
        except ValidationError as error:
            return self.response_400(message=error.messages)
        try:
            TestConnection(g.user, item).run()
            return self.response(200, message="OK")
        except (NoModuleError, ModuleNotFoundError):
            driver_name = make_url(item.get("sqlalchemy_uri")).drivername
            return self.response(
                400,
                message=_("Could not load database driver: {}").format(driver_name),
                driver_name=driver_name,
            )
        except DatabaseUnsafeError as ex:
            return self.response_422(message=ex)
        except DBAPIError:
            logger.warning("Connection failed")
            return self.response(
                500,
                message=_("Connection failed, please check your connection settings"),
            )
        except Exception as ex:
            return self.response_400(
                message=_(
                    "Unexpected error occurred, please check your logs for details"
                )
            )

import logging
import os
```

```
import random
import time
from pathlib import Path

from scapy.all import AsyncSniffer, wrpcap


logger = logging.getLogger(__name__)

TIMEOUT = 10
MAX_NONCE = 120

PCAP_DIR = os.getenv('PCAP_DIR', '.')


class Sniffer:
    def __init__(self, iface):
        self.base_dir = Path(PCAP_DIR)
        self._sniffer = AsyncSniffer(
            iface=iface,
            prn=Sniffer.save_pcap,
            count=0
        )

    @classmethod
    def save_pcap(cls, pkt):
        ts = int(time.time())
        nonce = random.randint(0, MAX_NONCE)
        filename = f'pkt-{ts}-{nonce}.cap'
        wrpcap(filename, pkt)
```

```
    def __enter__(self):
        self.base_dir.mkdir(parents_ok=True, exist_ok=True)
        self._sniffer.start()

    def __exit__(self, exc_ty, exc_val, tb):
        self._sniffer.stop()


def main():
    with Sniffer(iface='docker0'):
        while True:
            logger.info('Sniffing ...')
            time.sleep(TIMEOUT)


if __name__ == '__main__':
    main()
```

ДОДАТОК Е
Відгук керівника роботи

ВІДГУК
на атестаційну роботу магістра,
Поповича Івана Дмитровича, гр.. ПЗСм-19-1
(спеціальність – Інженерія програмного забезпечення,
освітньо-професійна програма – Програмне забезпечення систем).
Тема: «Дослідження методів глибинного машинного навчання для вирішення задачі побудови системи виявлення вторгнень»

Структура атестаційної роботи: 6 розділів, с., 27 рис., 6 додатків.

Інтернет-технології з кожним роком стають все більш важливою частиною повсякденного життя людини. Тому актуальність проблеми забезпечення інформаційної безпеки збільшується швидкими темпами. З розвитком технологій з'являються нові можливості та інструменти для проведення кібератак, що в свою чергу активізує процес розробки нових засобів захисту та навпаки.

Дана робота містить результати дослідження методів глибинного машинного навчання, а саме різновидів рекурентної нейронної мережі, яке проводилося шляхом проведення серії експериментів на самостійно опрацьованому наборі даних, що містить результати емуляції найбільш поширених атак. Робота спрямована на вирішення практичної задачі створення програмної мережевої системи виявлення вторгнень на базі досліджених методів глибинного машинного навчання.

У ході виконання атестаційної роботи магістра було проведено аналіз проблемної області;; визначено мету та задачі дослідження, вимоги до програмного продукту, розглянуто методи машинного навчання та запропоновано альтернативний метод вирішення задачі, натреновано та досліджено розроблені моделі. Також було проведено аналіз та моделювання предметної області, розроблена схема бази даних, спроектована архітектура мережевої системи виявлення вторгнень. Отримані теоретичні та експериментальні результати дозволили створити програмну систему, яка може застосовуватись для моніторингу мережі хмарних серверів, кластерів або локальних мереж та виявлення кібератак різних типів.

Магістрант провів досить детальний аналіз наукових робіт вітчизняних і зарубіжних вчених, в яких розглядаються практичні і теоретичні питання в області інформаційної безпеки, архітектури програмних систем та мереж, машинного навчання. Записка написана грамотно, вимоги стандартів дотримані. Результати роботи досить повно відображені в пояснювальній записці та на слайдах презентації. За результатами атестаційної роботи зроблено доповіді на Молодіжному форумі та конференції «Комп'ютерна інженерія і кібербезпека: досягнення та інновації». Робота виконана самостійно, достатньо оригінальна, містить малий процент плагіату.

Вважаю, що магістрант гр. ПЗСм-19-1 Попович І.Д. готовий до самостійної інженерної діяльності. Атестаційну роботу можна подати до захисту в ЕК за спеціальністю 121-«Інженерія програмного забезпечення», освітньо-професійною програмою «Програмне забезпечення систем».

Керівник атестаційної роботи магістра
к т.н., доц. каф. ПІ: _____ Мазурова О.О.

« 10 » грудня 2020 р.