

**МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ**

**ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І
СИСТЕМ**

**КВАЛІФІКАЦІЙНА РОБОТА
на тему:**

Розробка програмного забезпечення для моніторингу мережевої активності

Здобувача вищої освіти першого (бакалаврського) рівня,
4 курсу, групи КБ-19-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології (за галузями)»


(підпис)

Федорука Тараса Івановича
(прізвище, ім'я, по батькові)

Науковий керівник


(підпис)

Котух Євген Володимирович
к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Керівник освітньо-
професійної програми


(підпис)

Федотова - Півень Ірина
Миколаївна, к.т.н, доцент
(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Допущено до захисту за рішенням кафедри
протокол від «26» травня 2023 № 13

Завідувач кафедри


(підпис)

Філоненко Михайло Миколайович,
к.ф.-м.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене
звання)

Ірпінь – 2023

Реєстрація № 33

06.06.2023

(дата)



(підпис відповідальної особи)

Результати перевірки

до захисту

(до захисту, на доопрацювання)

29.05.2023

(дата)



(підпис)

Результат захисту

3 (задовільно)

(оцінка)

15.06.2023

(дата)



(підпис)

Голова комісії:



(підпис)

В.М. Франчук

Члени комісії:



(підпис)

М.М. Філоненко



(підпис)

О.В. Редич



(підпис)

Є.В. Котух

БЛАНК ОЦІНЮВАННЯ ВИПУСКНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії, секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст випускної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	23		15.06.2023 Франчук В.М. <i>В.М. Франчук</i> Філоненко М.М. <i>М.М. Філоненко</i>
2. Дотримання вимог щодо оформлення	від 0 до 10	7		Редич О.В. <i>О.В. Редич</i>
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	2		Котух С.В. <i>С.В. Котух</i> Суховерхня Л.П. <i>Л.П. Суховерхня</i>
4. Рецензент	від 0 до 5	3		
РАЗОМ БАЛІВ	від 0 до 50	35		

Оцінювання кваліфікаційної роботи науковим керівником - 30 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
65	D	3 (задовільно)	15.06.2023	Франчук В.М. <i>В.М. Франчук</i> Філоненко М.М. <i>М.М. Філоненко</i> Редич О.В. <i>О.В. Редич</i> Котух С.В. <i>С.В. Котух</i> Суховерхня Л.П. <i>Л.П. Суховерхня</i>

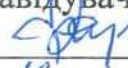
ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	
50-59	E	3 (задовільно)
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем
Галузь знань: 12 «Інформаційні технології»
Спеціальність: 122 «Комп'ютерні науки»
Освітня програма: «Інформаційні управляючі системи і технології (за галузями)»

Затверджую

Завідувач кафедри

 Філоненко М.М.
« 16 » 11 2022

Завдання на кваліфікаційну роботу
здобувачу вищої освіти
Федорука Тараса Івановича
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи :

«Розробка програмного забезпечення для моніторингу мережевої активності» затверджена наказом ректора від «15» листопада 2022 № 1205 (зі змінами наказ від «04» квітня 2023 № 361)

2. Строк здачі закінченої роботи «26» травня 2023

3. Мета роботи: розробка програмного модуля для системи управління подіями безпеки серверу сайту.

Об'єкт дослідження: захистом серверу вебсайту, від кібератак

Предмет дослідження: Системи керування захистом .

База дослідження:

4. Вихідні дані до роботи: Існуючі реалізації системи керування безпекою сайту.

5. Перелік графічного матеріалу: рисунки та графіки, формули, що пояснюють суть застосованих методів та засобів. Структурні схеми та таблиці, графіки результатів моделювання.

6. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом):

РОЗДІЛ 1 ПОСТАНОВКА ЗАДАЧІ ТА АКТУАЛЬНІСТЬ

1.1 Важливість інформаційних систем

1.2 Види мережових атак на інформаційні системи та APT-атаки

1.3 Проблема безпеки інформації у інформаційних системах

1.4 Методи виявлення атак

1.5 Firewall, як вид захисту від атак на інформаційну систему

1.6 Лог-файли, як спосіб контролю за станом системи та проблема збору даних

1.7 Аналіз принципів роботи та методів застосування IDS/IPS

1.8 Ефективність систем виявлення атак

РОЗДІЛ 2 АВТОМАТИЗАЦІЯ ПРОЦЕСУ ФОРМУВАННЯ ДАНИХ УПРАВЛІННЯ ГРАНИЧНИМ МІЖМЕРЕЖЕВИМ ЕКРАНОМ

2.1 Розробка пропозицій щодо автоматизації процесу формування даних

2.2 Налаштування IDS Suricata та написання власних правил

2.3 Структура правил для міжмережевого екрану

РОЗДІЛ 3 РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО МОДУЛЮ ФОРМУВАННЯ ДАНИХ УПРАВЛІННЯ IPS

3.1 Аналіз існуючих аналогів

3.2 Призначення програмного виробу та перелік техніко-економічних вимог.

3.3 Склад і характеристики потоків вхідної та вихідної інформації та основні вимоги

3.4 Запуск програми та інструкція з користування

3.5 Опис компонентів програми та вибір мови програмування

7. Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	Фактич
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 15.11.2022	15.11.2022
2	I розділ (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	01.02.2023	01.02.2023
3	II розділ (аналіз експериментальної частини про методики досліджень)	03.04.2023	04.04.2023
4	III розділ (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	10.05.2023	09.05.2023
5	Висновки	15.05.2022	15.05.2023
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	26.05.2023	26.05.2023
7	Отримання відгуку керівника	06.06.2023	06.06.2023
8	Отримання зовнішньої рецензії	08.06.2023	08.06.2023

8. Дата видачі завдання «16» листопада 2022

9. Керівник випускної кваліфікаційної роботи


(підпис)

Котух Є.В.
(прізвище, ініціали)

10. Керівник освітньо-професійної програми


(підпис)

Федотова-Півен
(прізвище, ініціали)

11. Завдання до виконання отримав здобувач вищої освіти


(підпис)

Федорук Т. І.
(прізвище, ініціали)

АНОТАЦІЯ

У дипломній роботі на тему " Розробка системи керування захистом вебсайту" було виконано роботу, яка має обсяг у 61 сторінок, включаючи 16 ілюстрацій, 1 таблицю, додаток та 8 бібліографічних посилань.

Метою цієї дипломної роботи було розробити програмний модуль для системи управління подіями безпеки вебсайту, а саме захисту серверів де базується вебсайт.

Для досягнення цієї мети в дипломній роботі були розглянуті загальні питання щодо актуальності розробки програмного модуля для системи керування подіями кібербезпеки. Було обґрунтовано вибір програмних та технічних засобів для створення модуля, а також проведено його тестування у віртуальній інформаційній системі.

В рамках дипломної роботи було представлено програмний модуль, який автоматизує процес створення правил для Firewall на основі індикаторів IDS та правил, визначених адміністратором безпеки.

Ключові терміни: системи IDS, кібербезпека, firewall, інформаційні системи, АРТ–атаки, програмний модуль.

ANNOTATION

The thesis on the topic "Development of a website security management system" has a work volume of 61 pages, including 16 figures, 1 table, an appendix and 8 bibliographic references.

By the method of this thesis, a software module was developed for the website security event management system, as well as the protection of the servers where the website is based.

To achieve this goal, general questions regarding the relevance of developing a software module for cyber security event management systems were posed in the thesis. The choice of software and technical means for creating the module was substantiated, as well as its testing in the virtual information system.

The thesis presented a software module that automates the process of creating rules for the Firewall based on IDS indicators and rules defined by the security administrator.

Key terms: IDS systems, cyber security, firewall, information systems, APT attacks, software module.

ЗМІСТ

ВСТУП	10
РОЗДІЛ 1	12
ПОСТАНОВКА ЗАДАЧІ ТА АКТУАЛЬНІСТЬ	12
1.1 Важливість інформаційних систем	12
1.2 Види мережових атак на інформаційні системи та 11	
1.3 Проблема безпеки інформації у інформаційних системах	14
1.4 Методи виявлення атак	15
1.5 Firewall, як вид захисту від атак на інформаційну систему	16
1.6 Лог-файли, як спосіб контролю за станом системи та проблема збору даних	17
1.7 Аналіз принципів роботи та методів застосування IDS/IPS	19
1.8 Ефективність систем виявлення атак	22
РОЗДІЛ 2	24
АВТОМАТИЗАЦІЯ ПРОЦЕСУ ФОРМУВАННЯ ДАНИХ УПРАВЛІННЯ ГРАНИЧНИМ МІЖМЕРЕЖЕВИМ ЕКРАНОМ	24
2.1 Розробка пропозицій щодо автоматизації процесу формування даних	24
2.2 Налаштування IDS Suricata та написання власних правил	25
2.3 Структура правил для міжмережевого екрану	27
РОЗДІЛ 3	28
РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО МОДУЛЮ ФОРМУВАННЯ ДАНИХ УПРАВЛІННЯ IPS	28
3.1 Аналіз існуючих аналогів	28
3.2 Призначення програмного виробу та перелік техніко-економічних вимог.	28
3.3 Склад і характеристики потоків вхідної та вихідної інформації та основні вимоги	29
3.4 Запуск програми та інструкція з користування	29
3.5 Опис компонентів програми та вибір мови програмування	30
3.6 Діаграми класів та створення віртуальної моделі інформаційної системи	32
3.7 Тестування програмного модулю	34
ВИСНОВКИ	40
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	41
ДОДАТОК	42

ВСТУП

Сучасний світ сильно покладається на інформаційні системи, які забезпечують автоматизацію і просування розвитку в різних галузях діяльності.

У питанні безпеки, деякі компоненти часто не отримують достатньої уваги, незважаючи на їх важливість. Багато людей вважають, що за допомогою належно налаштованого Firewall і антивірусного програмного забезпечення можна захистити себе від будь-яких загроз. Але насправді це недостатньо. Навіть найбільш захищені системи, сайти, сервери не є абсолютно невразливими. Експерти з комп'ютерної безпеки погоджуються з тим, що абсолютно безпечну систему створити неможливо.

Загрози інформаційній безпеці постійно зростають, особливо небезпечними є цільові або довгострокові АРТ-атаки зі складною структурою виконання. Це призводить до збільшення кількості систем, які призначені для захисту бізнесу від таких атак. У таких умовах важливо своєчасно виявляти інформаційні атаки та приймати комплексні заходи для їхнього припинення.

Адміністратори систем безпеки щодня отримують велику кількість повідомлень від різних сенсорів. Працездатність кожної підсистеми має велике значення для бізнесу в цілому, тому фахівці змушені аналізувати цей потік інформації, щоб попереджувати майбутні атаки.

Ця дипломна робота має на меті дослідження системи захисту в режимі реального часу. За останні 10 років кібератаки суттєво змінили свою природу та методи. Хоча базові заходи захисту, такі як антивіруси та файерволи, зробили інформаційні системи менш вразливими до типових атак, таких як віруси, DDOS-атаки та ARP-спуфінг, але АРТ-атаки потребують більш просунутих засобів боротьби, таких як системи управління подіями та безпекою інформації (SIEM).

Основною темою дослідження цієї роботи є процес автоматизації формування команд для актуаторів системи запобігання вторгненням (IPS) для системи SIEM в оперативному центрі безпеки (SOC). У цій роботі брандмауер виступає в якості актуатора IPS. Щоб краще розуміти роль SOC у мережі, наведено структурну схему типової мережі організації (рис. 1.2).

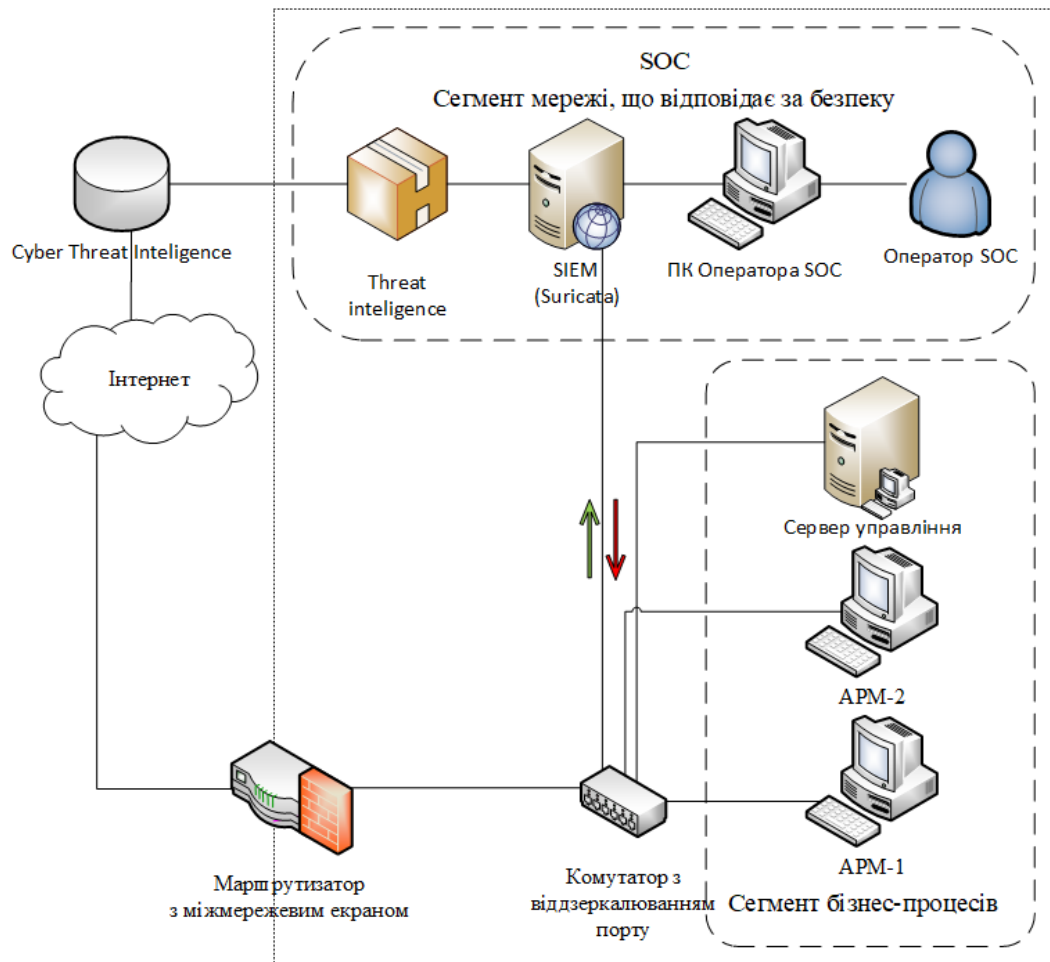


Рисунок 1.1 – Структурна схема віртуальної мережі.

Метою дипломної роботи є розробка програмного модуля для системи управління подіями безпеки серверу сайту, який буде використовуватися в рамках SIEM. Цей модуль буде аналізувати дані, зібрані з сенсорів IDS, і генерувати команди для актуаторів IPS.

Результатом роботи буде програмний модуль, який автоматизує процес формування списків контролю доступу на основі журналів подій IDS Suricata.

РОЗДІЛ 1

ПОСТАНОВКА ЗАДАЧІ ТА АКТУАЛЬНІСТЬ

1.1 Важливість інформаційних систем

У сучасному світі інформаційні системи відіграють важливу роль у різних сферах діяльності організацій. Вони призначені для задоволення потреб користувачів у інформації і можуть функціонувати як з використанням технічних засобів, так і без них. В залежності від ступеня автоматизації інформаційних процесів, інформаційні системи можуть бути класифіковані на ручні, автоматичні та автоматизовані. Ручні системи виконують всі операції вручну, без використання технічних засобів. Автоматичні системи здатні виконувати операції без участі людини, а автоматизовані системи включають в себе як людину, так і комп'ютерні технології. Сучасні інформаційні системи, як правило, автоматизовані, і їх використання широко застосовується у різних сферах управління. Вони можуть бути класифіковані за характером використання інформації та за сферою застосування.

1.2 Види мережових атак на інформаційні системи та АРТ-атаки

Розглянемо кілька небезпечних видів мережових атак. Один з поширених типів атак - переповнення буфера. Цей тип атаки часто використовується як складова різних шкідливих атак. Атаки переповнення можуть мати різні варіації, такі як атаки на стек або на форматування рядків. У випадку атаки на стек, зловмисники переповнюють буфер стеку, записуючи дані більшого обсягу, ніж передбачено програмою. В результаті

програма може виконати зловмисні інструкції, записані в стеку. Атака на форматування рядків також впливає на стек, але вона включає підміну адреси з одного на інший, що містить команди зловмисника.

"Пасивні" атаки, такі як використання сніфера, особливо небезпечні, оскільки вони складні для виявлення і можуть бути виконані з локальної мережі. Ці атаки дозволяють зловмисникам отримати різну інформацію безпосередньо з мережі, таку як дані про мережу, трафік сесій SSL або TLS, TCP і UDP, адреси IP або MAC, номери портів, імена користувачів, паролі.

Цільові атаки (APT-атаки) - це спрямовані на конкретні комерційні організації або державні відомства атаки. Зазвичай такі атаки не є масовими і плануються на тривалий період. Зловмисники досліджують інформаційні системи об'єкта, що підлягає атаці, дізнаються, яке програмне забезпечення використовується для різних цілей.

Об'єктами атаки можуть бути конкретні інформаційні системи або особи. Шкідливе програмне забезпечення спеціально розробляється для атаки, щоб уникнути виявлення стандартними антивірусами та іншими захисними засобами, які використовує об'єкт і досить добре вивчені зловмисниками.

Часто це використання недавно виявлених уразливостей. APT-атака складається з таких етапів:

- Збір інформації - зловмисні групи шукають та аналізують веб-сайти, домени, інформацію про співробітників, облікові записи співробітників, інформацію про партнерів, технічну документацію та іншу загальнодоступну інформацію.
- Дослідження мережі - цей етап включає створення мапи мережі, запис IP-адрес серверів, маршрутизаторів, комутаторів, DNS-серверів, пошук VPN-тунелів, хостів, проксі і т.д.

- Пошук вразливостей - на цьому етапі здійснюється пошук вразливостей на серверах, маршрутизаторах і т.д.
- Використання вразливостей - зловмисники зламують сервери, маршрутизатори або інше комп'ютерне обладнання, що знаходиться в мережі або підмережі цільової машини.
- Підміна облікових записів співробітників - це зазвичай використовується для фішингу і актуально для отримання інформації, логінів та паролів, особливо коли необхідно уникнути виявлення системами вторгнення, або коли не знайдені вразливості в мережевій інфраструктурі.
- Фізичне проникнення - цей етап рідко використовується, коли шукана інформація не має електронного варіанту або комп'ютер, що містить секретну інформацію, ізольований від мережі, а також для отримання фізичного доступу до комп'ютера.
- Утримання в системі - зловмисник закріплюється в системі, щоб мати довгостроковий доступ до інформації.

1.3 Проблема безпеки інформації у інформаційних системах

Захист інформаційних ресурсів від негативного впливу, що може призвести до зниження їх цінності, є важливим завданням для організацій, підприємств і держав. Цей вплив називається загрозою, а захист полягає у створенні і підтримці систем, які запобігають або ускладнюють реалізацію цих загроз та зменшують можливі збитки. Захист інформації орієнтований на забезпечення безпеки оброблюваної інформації та інформаційних систем, забезпечуючи збереження їх властивостей.

Кількість кіберзлочинів, які вчиняються через комп'ютерні мережі, постійно зростає, особливо з поширенням мобільних технологій та Інтернету.

Організаційні заходи можуть вирішити значну частину проблем із захистом інформації в інформаційних системах. Однак, з розвитком технологій і збільшенням обсягу інформації, зростає потреба в застосуванні автоматизованих засобів управління вразливостями в інформаційних системах.

Згідно з аналізом Центру скарг щодо Інтернет-злочинності за 2017 рік, було отримано 262 813 скарг про вторгнення до інформаційних мереж, при цьому користувачі та організації зазнали збитків на суму у розмірі 781 841 611 доларів.

Україна також спостерігає зростання кількості кіберзлочинів. За даними Державної служби статистики України, у 2016 році було зареєстровано 5400 злочинів у сфері ІТ, а в 2017 році - 7 025.

1.4 Методи виявлення атак

Аналіз інформації, отриманої в результаті, є важливим елементом систем виявлення вторгнень. Виявлення зловмисних дій може здійснюватись двома методами: виявленням аномалій і виявленням зловживань.

Метод виявлення аномалій ґрунтується на моделях передбачуваної поведінки користувачів і додатків, які розпізнають відхилення від "нормальної" поведінки і вважають їх потенційними порушеннями безпеки. Цей метод базується на відмінностях між вторгненнями та нормальною системною поведінкою. Для виявлення аномалій створюються профілі, що описують типову роботу користувачів, хостів або мережевих з'єднань і можуть бути достатньо точними. Наприклад, визначаються звичні дії конкретного користувача (реєстрація в системі в певний час, перегляд електронної пошти, транзакції з базами даних, активність поза робочим часом тощо). Якщо система виявляє дії, які не відповідають встановленому профілю (наприклад, реєстрація в системі в нічний час, використання засобів

компіляції та налагодження, помилки при доступі до файлів), ця діяльність вважається підозрілою. Однак методи виявлення аномалій і системи виявлення вторгнень, що базуються на них, часто спричиняють багато помилкових спрацьовувань через нечіткі межі "нормальної" поведінки. Незважаючи на це, методи виявлення аномалій мають потенціал для виявлення нових форм атак і можуть бути корисними у змінних середовищах.

Методи виявлення зловживань базуються на порівнянні дій системи, подій або їх комбінацій з попередньо відомими зразками, що описують відомі атаки. Якщо зразок відповідає відомій атаці, його визначають як сигнатуру. Використання методів виявлення зловживань є досить ефективним для виявлення атак і при цьому зазвичай не призводить до багатьох помилкових спрацьовувань тривог. Швидке виявлення зловживань дозволяє адміністраторам системи вчасно прийняти заходи щодо забезпечення безпеки. Однак цей підхід має недоліки, такі як постійне оновлення бази даних з новими сигнатурами, оскільки детектори можуть аналізувати тільки відомі сигнатури.

Після отримання та аналізу інформації про подію, система генерує звіти у стандартному форматі або автоматично виконує дії, такі як блокування портів чи програм. Також можуть бути застосовані більш активні відповіді, наприклад, надсилання повідомлення на телефон або включення попереджувального звукового сигналу. Активні дії можуть включати перенастроювання конфігурації роутера для блокування IP-адреси зловмисника або організацію відповідного контратаку. Однак такі заходи мають певний ризик, оскільки можуть спрямовуватись проти невинних осіб, оскільки зловмисники часто використовують підроблені IP-адреси. Наприклад, у разі вторгнення з використанням фальсифікованого трафіку з підробленою IP-адресою, система може повідомити адміністратора про атаку.

1.5 Firewall, як вид захисту від атак на інформаційну систему

Міжмережевий екран (firewall) є програмним засобом, який забезпечує контроль за вхідною та вихідною інформацією в інформаційній системі. Він виступає захисною стіною між локальною мережею та зовнішніми мережами, маючи на меті запобігти можливим загрозам. Міжмережевий екран контролює вхідний та вихідний мережевий трафік, дозволяючи блокувати різноманітні мережеві атаки, запобігати передачі інформації про комп'ютер іншим серверам та унеможливлювати дію програм-троянів і засобів віддаленого адміністрування.

Робота міжмережевого екрана полягає у аналізі структури і вмісту інформаційних пакетів, які надходять з зовнішньої мережі. Залежно від результатів аналізу, екран пропускає пакети до внутрішньої мережі або повністю їх відфільтровує. Існують два типи міжмережевих екранів: апаратні та програмні.

Апаратний тип представляє собою фізичний пристрій, що підключається до мережі. Він відстежує всі аспекти обміну даними, перевіряє адреси джерела та призначення кожного повідомлення, що обробляється. Це допомагає забезпечити безпеку та запобігти небажаним проникненням в мережу чи комп'ютер.

Програмний тип міжмережевого екрана виконує ті ж функції, але використовує програмний продукт, що працює на комп'ютері або шлюзі. Цей тип реалізації є більш поширеним.

Мережеві екрани можуть працювати на різних рівнях протоколів моделі OSI. На рівні мережі вони фільтрують вхідні та вихідні пакети за IP-адресами. На рівні транспорту фільтрація відбувається за номерами портів TCP і прапорцями, які містяться в пакетах. На рівні додатків виконується аналіз прикладних протоколів і контроль за змістом потоків даних.

1.6 Лог-файли, як спосіб контролю за станом системи та проблема збору даних

Збір і аналіз логів у системі є важливим для контролю за роботою та станом системи. Лог-файл - це спеціальний файл, який містить службову та статистичну інформацію про події в системі або програмі. Багато операційних систем і серверного програмного забезпечення мають розширену систему логування, яка дозволяє реєструвати різні типи подій у відповідних лог-файлах.

Лог-файли містять сирі дані, які потребують подальшої обробки. Якість обробки визначає якість отриманої статистики. В лог-файлах зберігається вся необхідна інформація, яка дозволяє отримати повне розуміння про користувачів та клієнтів інформаційної системи.

Важливо зазначити, що отримати абсолютно точну статистику практично неможливо через технічні обмеження. Немає жодного коректного способу оцінити точність вимірюваних показників порівняно з реальністю. Однак прийнято вважати, що середні відхилення не перевищують 5-10%.

Для ефективного виявлення та аналізу зловмисних дій необхідно мати детальні дані про роботу системи. Багато операційних систем забезпечують різні види аудиту, які дозволяють створювати журнали реєстрації для різних користувачів. Журнали реєстрації можуть бути налаштовані для реєстрації подій, пов'язаних з безпекою, або для повного відображення всіх системних викликів, що ініціюються процесами.

Мережеві екрани та маршрутизатори також реєструють події в журналах реєстрації, які містять інформацію про створення мережевих з'єднань або записи про передачу кожного пакета.

Збір та зберігання інформації для подальшого аналізу є компромісом між витратами та ефективністю системи виявлення вторгнень. Конфігурація системи, в якій всі події докладно записуються в лог-файли, вимагає великої

обчислювальної потужності хоста та значного обсягу дискового простору, що призводить до збільшення витрат. Збір та зберігання необхідної інформації є важливим, але витратним завданням. Питання щодо того, яку інформацію слід реєструвати та де її зберігати, залишається відкритим.

1.7 Аналіз принципів роботи та методів застосування IDS/IPS

Після появи нового типу атаки, відомого як АРТ, з'явилася необхідність у використанні систем кібернетичного захисту, які працюють у режимі реального часу. Для розв'язання проблеми захисту від складних атак було введено практику створення центрів безпеки, так званих Security Operation Center (SOC, рис. 1.2), де здійснюється керування безпекою інформаційної мережі організації. Усередині SOC працюють системи SIEM, які аналізують інформацію з різних індикаторів у мережі, таких як файерволи, антивіруси, системи IDS/IPS, журнали операційних систем, і можуть попередити про можливу атаку на інформаційну систему. Крім того, SOC включає інтелектуальну базу даних про вразливості та загрози, відому як Cyber Threat Intelligence.

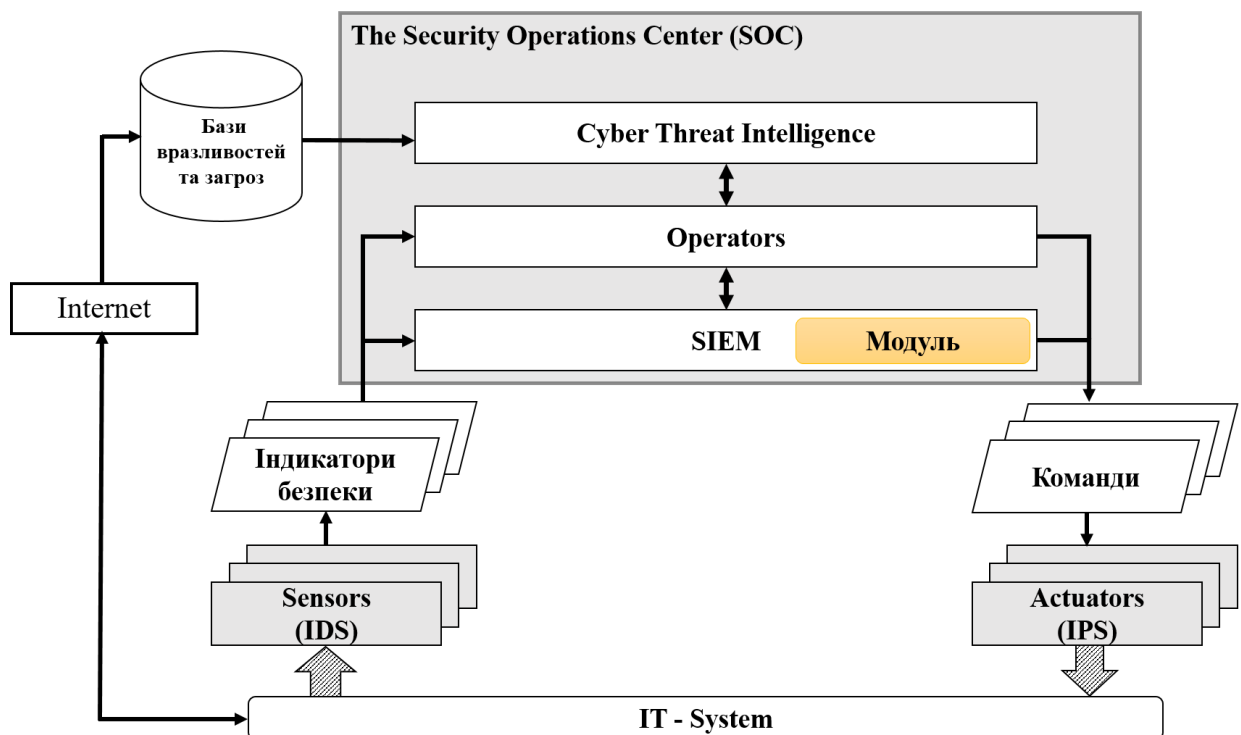


Рисунок 1.2 – Структура системи кіберзахисту в режимі реального часу.

Головна особливість цієї системи захисту полягає у її комплексному підході та розгляді різних аспектів безпеки, що відрізняє її від базових інструментів захисту.

По-перше, слід зазначити, що ця система кібернетичного захисту працює в межах SOC (Центру операцій з безпеки). SOC - це спеціальна команда інформаційної безпеки, яка відповідає за моніторинг та аналіз безпекового стану організації. Вона використовує системи моніторингу та управління інформаційною безпекою (SIEM) та інтелектуальні бази даних про вразливості (Cyber Threat Intelligence), спеціально налаштовані для даної інформаційної системи з урахуванням контексту (обладнання, програмне забезпечення і т. д.) та постійної загрози.

Головною метою команди SOC є виявлення, аналіз та реагування на кібербезпекові інциденти. На центрах з безпеки операцій зазвичай працюють безпекові аналітики, інженери та менеджери, які відповідають за стан системи. Співробітники SOC співпрацюють з командами реагування на інциденти, щоб забезпечити швидке врегулювання питань безпеки після їх виявлення.

Основна мета використання інтелектуальної бази знань про вразливості та загрози системи - допомогти організаціям зрозуміти ризики найпоширеніших та серйозних зовнішніх загроз, таких як атаки нульового дня, АРТ-атаки та експлойти. Хоча загрози також можуть бути внутрішніми, акцент робиться на типи загроз, які можуть бути використані ззовні.

Головна мета SIEM як програмного продукту - збирати дані з сенсорів IDS (компонентів інформаційної системи) у вигляді лог-файлів, їх нормалізація та аналіз. Результатом аналізу є формування команд для активаторів IPS (брандмауерів).

Отже, операційні центри безпеки здійснюють моніторинг та аналіз активності в мережах, серверах, кінцевих точках, базах даних, програмах, веб-сайтах та інших системах, з метою виявлення аномальної діяльності, яка може свідчити про безпековий інцидент. SOC відповідає за ідентифікацію, аналіз, захист, розслідування та повідомлення потенційних безпекових інцидентів.

У класичному випадку команди для активаторів IPS формують член команди реагування SOC. Аналіз інформації з індикаторів займає багато часу, тому цей процес зазвичай є довготривалим.

Метою дослідження є розробка та тестування програмного модуля, що автоматизує процес формування команд для активатора IPS (брандмауера). Зростання обсягів інформації, потреба в швидкому її обробці та складному аналізі приводять до збільшення кількості вразливостей в інформаційних системах. Кожен засіб захисту спрямований на певну безпечну загрозу в системі і має свої переваги та недоліки. Тільки поєднання різних засобів захисту може забезпечити максимальний рівень захисту від широкого спектру атак.

Для контролю за станом інформаційної системи та оброблюваною в ній інформацією кожна подія фіксується у вигляді записів у лог-файлі.

Для аналізу та реєстрації подій, пов'язаних з передачею інформації у мережі, розроблено системи виявлення вторгнень (IDS/IPS). IDS/IPS системи служать для моніторингу системи та прийняття рішень на основі проаналізованих подій, визначаючи їх як підозрілі чи ні. Вони можуть виявити дії зловмисника або шкідливе програмне забезпечення, які обійшли мережевий екран, і повідомити адміністратора мережі, який прийматиме заходи для запобігання атаці.

IDS/IPS системи є програмними або апаратними засобами, які призначені для виявлення незаконного доступу. Вони автоматизують процес перегляду подій, що виникають у комп'ютерній системі або мережі, та аналізують їх з точки зору безпеки.

Виявлення проникнень та аномальної мережевої активності дозволяє підприємствам оперативно реагувати та застосовувати відповідні заходи безпеки. Системи виявлення вторгнень (IDS) забезпечують виявлення мережевих атак, підвищення прав користувачів, неавторизованого доступу до важливих файлів та дій шкідливого програмного забезпечення. Використання

IDS допомагає досягти кількох цілей, включаючи виявлення вторгнень, контроль адміністрування, прогнозування майбутніх атак, отримання інформації про проникнення та визначення джерела атак.

Розгортання системи виявлення проникнень є ефективним заходом, оскільки дозволяє виявити проникнення зловмисника в систему та швидко реагувати. IDS дозволяє визначити сигнатури атак, проводити зондування та виявляти підозрілі дії. Наявність IDS ускладнює завдання зловмисникам, які намагаються знайти уразливості у системі.

Основними компонентами систем виявлення вторгнень є сенсорна підсистема, підсистема аналізу, сховище, консоль управління та модуль реагування. Сенсорна підсистема збирає події, пов'язані з безпекою мережі або системи. Підсистема аналізу виявляє мережеві атаки та підозрілі дії. Сховище зберігає базу подій та результати аналізу. Консоль управління дозволяє конфігурувати систему, спостерігати за станом мережі та переглядати виявлені інциденти. Модуль реагування виконує інструкції по протидії несанкціонованому вторгненню.

Ці системи виявлення вторгнень допомагають забезпечити безпеку мереж та систем, виявляти атаки та попереджати їхні наслідки.

1.8 Ефективність систем виявлення атак

Більшість систем виявлення вторгнень, як от відкриті IDS Snort і Suricata, або комерційна Stonesoft, StoneGate, базуються на методах визначення зловживань шляхом аналізу трафіку, що проходить через мережу. Однак, ці системи потребують постійного оновлення баз даних з новими сигнатурами, оскільки вони використовують лише відомі моделі сигнатур для аналізу. Водночас, системи, які використовують методи виявлення аномалій, можуть породжувати багато хибних тривог, оскільки немає чітких меж для «нормальної» поведінки користувачів і системи.

Існує ймовірність того, що системи виявлення вторгнень, що базуються на визначенні аномалій, здатні виявляти нові форми атак, навпаки до систем, що використовують сигнатури з баз даних. Деякі фахівці рекомендують поєднувати обидва методи виявлення, проте це потребує подальших досліджень.

Крім виявлення атак, системи виявлення вторгнень повинні мати здатність аналізувати великі обсяги інформації, що генерується в швидкісних мережах та продуктивних системах. За рахунок зростання обчислювальної потужності комп'ютерів та використання гігабітних мереж, аналіз великих обсягів даних стає викликом. Існують різні методи для вирішення цієї проблеми, такі як поділ потоку подій та використання периферійних сенсорів.

Підходу поділу потоку подій використовує модуль секатора (SLICER), який розбиває потік інформації на частини для подальшого аналізу сенсорами в реальному часі. Однак, цей підхід має вимоги до модуля поділу, оскільки він повинен гарантувати виявлення всіх сценаріїв атак. Неправильний поділ потоку може призвести до втрати або недостатчі необхідних даних для аналізу сенсорами.

Використання периферійних сенсорів передбачає розміщення датчиків на хостах для природного розділення трафіку. Проте, управління та координація багатьох сенсорів є складним завданням. Важливо правильно розміщувати сенсори, зокрема вузлах мережі для виявлення атак, заснованих на топології мереж.

Управління системою сенсорів також є важливим аспектом, оскільки обчислювальні мережі постійно розвиваються, а загрози стають все більш складними. Система сенсорів повинна адаптуватися до змін відповідно до цих еволюційних змін.

РОЗДІЛ 2

АВТОМАТИЗАЦІЯ ПРОЦЕСУ ФОРМУВАННЯ ДАНИХ УПРАВЛІННЯ
ГРАНИЧНИМ МІЖМЕРЕЖЕВИМ ЕКРАНОМ

2.1 Розробка пропозицій щодо автоматизації процесу формування даних

На сьогоднішній день аналіз індикаторів безпеки вимагає значної кількості часу, оскільки він проводиться вручну оператором безпеки. З метою автоматизації процесу формування даних управління було запропоновано рішення даної проблеми. Пропонується розробити програмний модуль, який буде інтегрований у SIEM і використовувати індикатори з IDS Suricata. Цей модуль зможе автоматично створювати правила управління для міжмережевого екрана, замінюючи ручне створення. Для кращого розуміння структури та принципів роботи цього програмного модуля, була розроблена та описана технологія обробки інформації, що використовується (див. рис. 2.2, табл. 2.1).

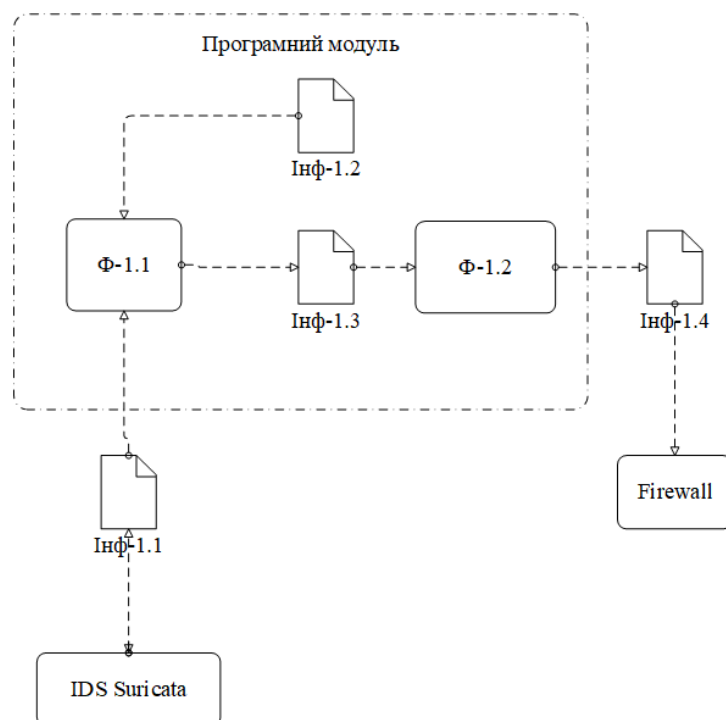


Рисунок 2.2 – Інформаційно-функціональна структура процесу автоматизації

Таблиця 2.1 – Опис інформаційно-функціональної структури

№	Код	Найменування	Примітка
Функції технології			
1	Ф-1.1	Аналіз індикаторів безпеки IDS Suricata	
2	Ф-1.2	Формування команд для міжмережевого ерану	
Проміжна інформація			
1	Інф-1.1	Лог-файли, що містять індикатори безпеки IDS Suricata	
2	Інф-1.2	Правила програмного модулю, задані членом команди реагування	
3	Інф-1.3	Індикатори безпеки, що підпадають під правила задані у модулі	
4	Інф-1.4	Команди для міжмережевого екрану	

Член команди реагування обирає лог-файл IDS Suricata за визначений період часу. Програмний модуль аналізує цей лог-файл згідно зі структурою, а потім порівнює отримані результати з правилами, заданими в програмному модулі.

У випадку, якщо правило виконується, програмний модуль формує команду для Firewall згідно з шаблоном, використовуючи дані з лог-файла, який відповідає правилам, заданим в програмному модулі.

2.2 Налаштування IDS Suricata та написання власних правил

Для формування команд актуаторам необхідно отримати вхідні дані, використовуючи лог-файли з IDS Suricata. Для цього потрібно ознайомитися з правилами та процесом налаштування IDS Suricata. Основний конфігураційний файл IDS Suricata має назву `suricata.yaml` і знаходиться за шляхом `/etc/suricata/suricata.yaml`. В цьому файлі виконується підключення правил для IDS та їх налаштування. Усі правила даного IDS розміщені за шляхом `/etc/suricata/rules/`. Для кожного протоколу існує окремий файл з правилами. IDS Suricata використовує певний формат для створення правил (див. рис. 2.1):

```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET
TROJAN Likely Bot
Nick in IRC (USA +..)"; flow:established,to_server;
flowbits:isset,is_proto_irc; content:"NICK "; pcre:"/NICK
.*USA.*[0-9]{3,}/i"; classtype:trojan-activity;
reference:url,doc.emergingthreats.net/2008124;
reference:url,www.emergingthreats.net/cgi-
bin/cvswweb.cgi/sigs/VIRUS/TROJAN_IRC_Bots;
sid:2008124; rev:2;)
```



Рисунок 2.1 – Формат запису правил IDS suricata.

Правило IDS Suricata складається з дії (оповіщення, пропуск, пропуск з попередженням або відкидання), заголовка (протокол, адреса призначення та адреса відправника) та параметрів правила. Файли з індикаторами знаходяться за шляхом `/var/log/suricata` і містяться у таких файлах:

- `suricata.log`: містить інформацію про старт програми
- `stats.log`: містить статистику про трафік мережі

- fast.log: містить інформацію про підозрілий трафік
- eve.json: містить інформацію про трафік мережі у форматі JSON Для запуску роботи необхідно виконати команду:

`suricata -c /etc/suricata/suricata.yaml -i eth0` Розглянемо структуру лог-файлу IDS Suricata. Запис індикатору безпеки містить таку корисну для нас інформацію: дату створення запису, протокол, повідомлення, адресу відправника та отримувача, пріоритет.

2.3 Структура правил для міжмережевого екрану

Оскільки IDS Suricata працює на ОС Linux, в якості файрволу оберемо вбудований в ОС UFW. Ядро Linux має підсистему Netfilter (мережевий фільтр), який використовується для управління проходженням і фільтрацією мережевих пакетів. Утиліта iptables є інтерфейсом до системи Netfilter і має розширені можливості з налаштування мережевих фільтрів, але вважається складною у використанні. UFW - це інструмент, який дозволяє легко налаштувати правила iptables [5]. Нижче наведено формат створення правил для UFW:

- `ufw allow|deny|reject|limit [in|out on INTERFACE] [log|log-all] [proto protocol] [from ADDRESS [port PORT]] [to ADDRESS [port PORT]]`
Правило складається з дії, типу трафіку з зазначенням інтерфейсу, можливості логування, протоколу, адреси відправника та отримувача. Щоб активувати файрвол, введіть наступну команду: `sudo ufw enable`

РОЗДІЛ 3

РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО МОДУЛЮ ФОРМУВАННЯ ДАНИХ УПРАВЛІННЯ IPS

3.1 Аналіз існуючих аналогів

Після проведення аналізу доступної літератури та досліджень, встановлено, що подібні модулі відсутні на сучасному етапі розвитку. Тому можна вважати дану розробку унікальною. Застосування даного програмного модуля має великий потенціал для вирішення проблеми автоматизації процесу формування даних управління для мережевого екрану.

3.2 Призначення програмного виробу та перелік техніко–економічних вимог.

Програмний модуль, що називається "SIEM Module", розроблений для автоматичного формування команд для фаєрволу Uncomplicated Firewall (UFW). Модуль включає такі функціональні можливості: аналіз лог-файлів системи виявлення інтруксів IDS Suricata, створення правил для використання в модулі та їх зберігання у файлі, аналіз логів відповідно до заданих правил в модулі та формування команд для фаєрволу в разі виконання цих правил.

Завдяки безкоштовності розробки даного програмного модулю, його економічна ефективність буде високою. Він може значно заощадити час оператора безпеки, що становить приблизно 40% робочого часу. Таким чином, можна стверджувати, що програмний модуль допоможе зекономити до 40% витрат на заробітну плату оператора, що є значною економічною вигодою.

3.3 Склад і характеристики потоків вхідної та вихідної інформації та основні вимоги

Вхідні дані для програмного модулю є лог-файли з індикаторами IDS Suricata. У модулі існує єдиний канал, через який проходить інформація та який використовується для перетворення даних в команди для міжмережевого екрану. Цей процес відбувається після аналізу індикаторів безпеки. Вихідна інформація представляється у форматі команд для міжмережевого екрану UFW.

Для коректного запуску програмного модулю на ЕОМ рекомендується мати двоядерний процесор з тактовою частотою 1.5GHz або вище, а також оперативну пам'ять обсягом 1GB DDR3 або більше. Відеокарта може бути будь-якою. Також необхідно мати вільне місце на жорсткому диску не менше 2GB для установки програмного модулю. Для роботи програми потрібно мати встановлену Java Virtual Machine версії 8.x або новіше.

3.4 Запуск програми та інструкція з користування

Встановлення операційної системи Ubuntu 16.04 на персональний комп'ютер можна здійснити автоматично без необхідності додаткових дій. Для встановлення IDS Suricata, вам необхідно перейти на офіційний веб-сайт suricata-ids.org та слідувати інструкціям з встановлення, що надаються на цьому сайті.

Щоб встановити Java Development Kit (JDK) та Java Virtual Machine (JVM), вам потрібно перейти на відповідний веб-сайт і дотримуватись інструкцій з встановлення, які надаються на цьому сайті.

Для використання програми спочатку потрібно налаштувати правила для програмного модуля, відповідно до встановленої форми. Потім виберіть лог-файл з IDS Suricata для його аналізу. Програмний модуль проводить аналіз

лог-файлу і, якщо записи лог-файлу відповідають заданим правилам в програмному модулі, генерується команда для UFW. В іншому випадку нічого не відбувається. Для кращого розуміння наведена блок-схема з алгоритмом роботи (рис. 3.1).

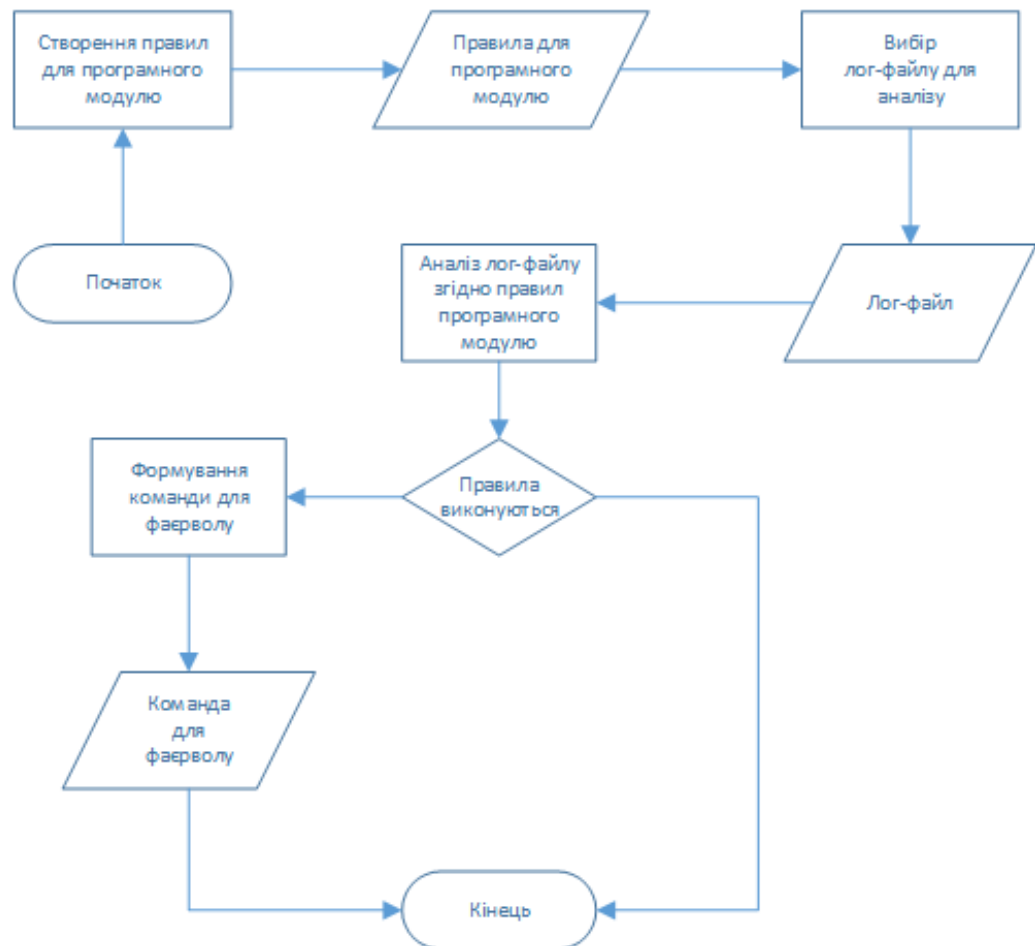


Рисунок 3.1 – Алгоритм роботи у вигляді блок-схеми.

3.5 Опис компонентів програми та вибір мови програмування

Програмний модуль складається з наступних компонентів:

Дві сутності об'єктів, що представляють індикатори в лог-файлах IDS Suricata та правила, задані в програмному модулі. Сутність індикатору містить наступні поля: час запису, протокол, повідомлення, IP-адреса відправника, порт відправника, IP-адреса отримувача, порт отримувача.

Сутність правила модулю містить наступні поля: дата від якої аналізуються індикатори, дата до якої аналізуються індикатори, ключові слова в індикаторі, протокол, IP-адреса відправника, порт відправника, IP-адреса отримувача, порт отримувача.

Аналізатор індикаторів, що перевіряє їх поля з правилами в модулі.

Парсер лог-файлів IDS Suricata, що перетворює кожен запис у сутність індикатору для подальшого аналізу з використанням регулярних виразів.

Менеджер правил модулю, відповідальний за збереження правил під час роботи модулю та може зберігати правила у файлі, додавати нові правила або видаляти їх.

Модуль формування правил, що створює команди для UFW згідно шаблону команд.

Кросплатформність - це здатність програмного забезпечення працювати на різних платформах та операційних системах. В сучасному ринку все більше популярності набувають кросплатформні програми, які можуть працювати на Windows, Linux, FreeBSD і MacOS. У випадку даного програмного модулю, вирішено, що він буде підтримувати операційну систему Ubuntu 16.04.

Головним принципом кросплатформних додатків є чітке розділення інтерфейсу програми та її логіки роботи. Крім того, важливо використовувати широко визнані стандарти для забезпечення максимальної сумісності.

Java є однією з найпопулярніших мов програмування, орієнтованих на безпеку, мобільність та надійність. Вона широко використовується для різних цілей і є найкращим вибором для створення програмного модулю. У даному випадку рекомендується використовувати Java 8 з бібліотекою JavaFx.

3.6 Діаграми класів та створення віртуальної моделі інформаційної системи

Для кращого розуміння структури програмного модулю та його компонентів, можемо ознайомитися з діаграмами класів (див. рис. 3.2).

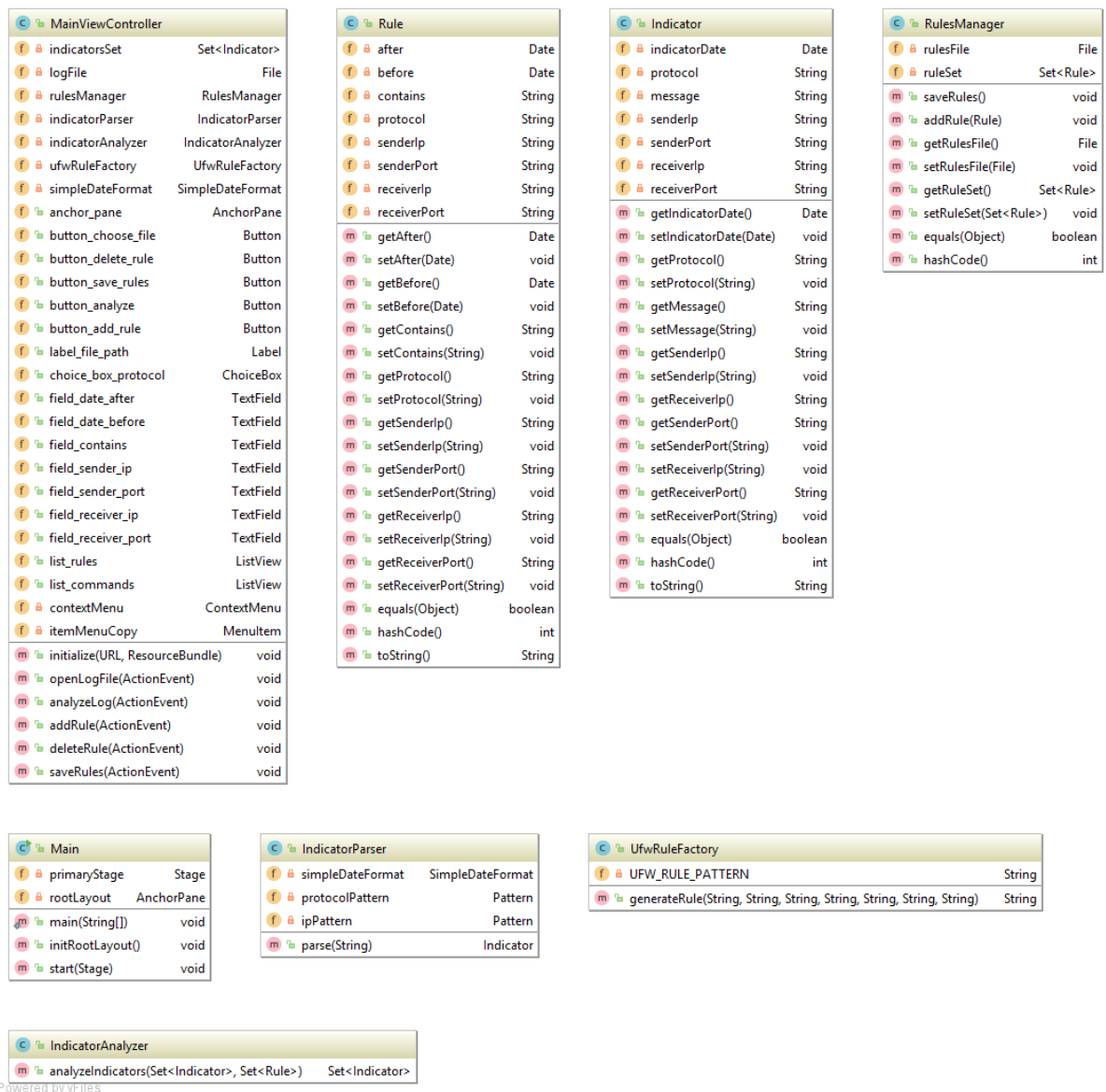


Рисунок 3.2 – Діаграми класів.

Опишемо класи програмного модулю:

- Клас Main відповідає за запуск програмного модулю.

- Клас `MainViewController` відповідає за зв'язок з графічним інтерфейсом.
- Класи `Indicator` та `Rule` представляють сутності індикаторів та правил модулю.
- Клас `IndicatorParser` формує об'єкти класу `Indicator` з записів лог-файлу.
- Клас `IndicatorAnalyzer` проводить аналіз індикаторів згідно правил, заданих у програмному модулі.
- Клас `UfwRuleFactory` формує команди для фаєрволу відповідно до інформації з індикаторів, які підпадають під правила після аналізу.

Створення інформаційної системи буде проводитись у віртуальній мережі, яка буде створена за допомогою VMware Workstation Pro.

Використання цього програмного забезпечення дозволить:

- Легко налаштувати необхідну топологію віртуальної мережі.
- Налаштувати вбудований DHCP-сервер.
- Гнучко налаштувати хости відповідно до вимог їх функціоналу.

Віртуальна локальна комп'ютерна мережа є групою хостів, які взаємодіють між собою, незалежно від їх фізичного розташування, і вона має ті самі атрибути, що й фізична локальна мережа. Застосування віртуальної мережі дозволяє згрупувати кінцеві станції разом, навіть якщо вони не знаходяться на одному мережевому комутаторі. Віртуальна мережа також дозволяє реконфігурувати мережу за допомогою програмного забезпечення, замість фізичного переміщення пристроїв.

3.7 Тестування програмного модулю

Для з'єднання комп'ютерів потрібне мережеве обладнання, яке складається з двох видів:

- Активне мережеве обладнання, яке обробляє або перетворює передавану по мережі інформацію. Це можуть бути мережеві карти, маршрутизатори, принт-сервери.
- Пасивне мережеве обладнання, яке просто передає сигнал на фізичному рівні. Сюди входять мережеві кабелі, конектори, мережеві розетки, повторювачі і підсилювачі сигналу.

Для створення програмної моделі автоматизації процесів управління вразливостями інформаційних систем необхідні такі компоненти:

- Міжмережевий екран.
- Сервер, на якому працює IDS Suricata.
- АРМ мережі.
- Сервер веб-сайту.
- Комутатор.

Для тестування програмного модулю була розгорнута локальна мережа, яка складається з Web-сервера, сервера IDS Suricata та АРМ.

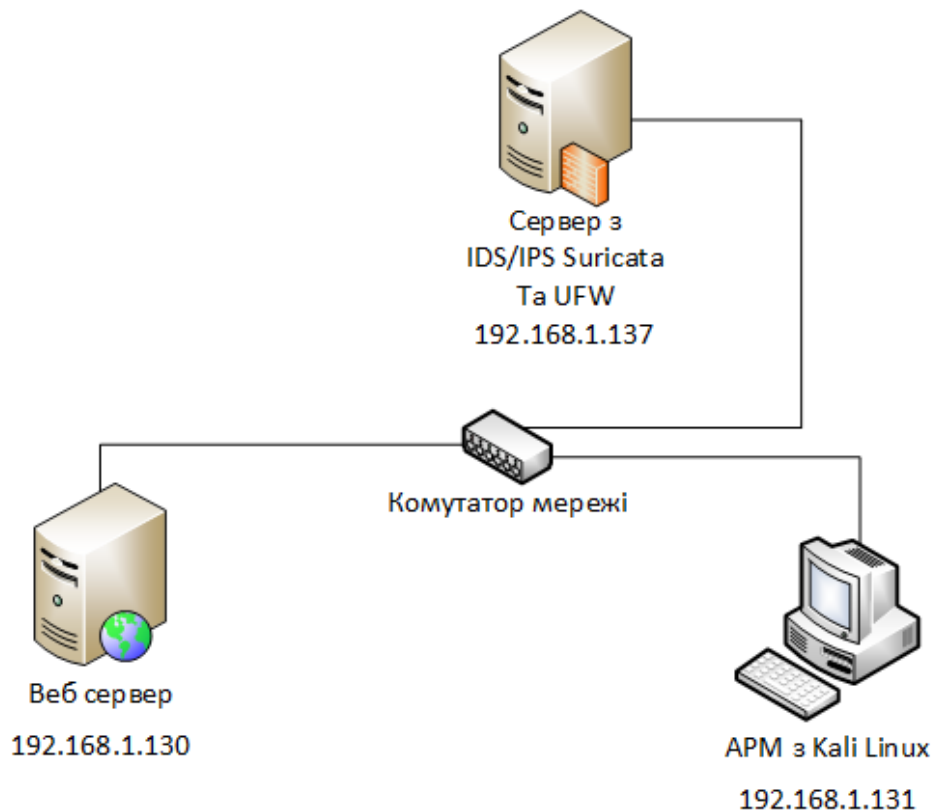


Рисунок 3.3 – Схема віртуальної мережі.

На наступному кроці ми визначимо набір правил у файлі `/etc/suricata/rules/test.rules` для IDS Suricata. Ці правила будуть спрямовані на виявлення мережевої атаки SYN-флуду. Після цього ми підключимо ці правила у конфігураційному файлі `suricata.yaml`.

```
alert udp any any -> $HOME_NET 80 (msg:"LOCAL DOS UDP port 80 flood inbound, Potential DOS"; threshold: type both, track by_dst, count 70, seconds 5; classtype:misc-activity; sid:3;)
alert udp $HOME_NET any -> any 80 (msg:"LOCAL DOS UDP port 80 flood outbound, Potential DOS"; threshold: type both, track by_dst, count 70, seconds 5; classtype:misc-activity; sid:4;)
alert tcp any any -> any any (msg:"LOCAL DOS SYN packet flood inbound, Potential DOS"; flow:to_server; flags: S,12; threshold: type both, track by_dst, count 70, seconds 5; classtype:misc-activity; sid:5;)
alert tcp $HOME_NET any -> any any (msg:"LOCAL DOS SYN packet flood outbound, Potential DOS"; flow:to_server; flags: S,12; threshold: type both, track by_dst, count 70, seconds 5; classtype:misc-activity; sid:6;)
```

Рисунок 3.4 – Власні правила для IDS Suricata.

SYN-флуд - це один з видів мережесих атак, що входить до категорії атак типу відмова в обслуговуванні (DoS). В цьому типі атаки зловмисник відправляє велику кількість SYN-запитів (запити на підключення за протоколом TCP) протягом короткого періоду часу.

Суть атаки полягає у тому, що зловмисник переповнює чергу на підключення на сервері, надсилаючи SYN-запити. При цьому він може ігнорувати SYN + ACK пакети, які надсилає цільовий сервер у відповідь, або підробляти заголовок пакета таким чином, що відповідь SYN + ACK відправляється на неіснуючу адресу. Це призводить до появи напіввідкритих з'єднань (Half-open connections) в черзі підключень, які очікують підтвердження від клієнта. Після певного тайм-ауту ці підключення відкидаються.

Метою зловмисника є підтримувати чергу заповненою, щоб запобігти новим підключенням. Це призводить до того, що легітимні клієнти не можуть встановити з'єднання або роблять це з помітною затримкою. Зараз ми запустимо IDS (рис. 3.5).

```
administrator@ubuntu:~$ sudo suricata -c /etc/suricata/suricata.yaml -i eth0
[sudo] password for administrator:
1/7/2018 -- 07:24:04 - <Notice> - This is Suricata version 4.0.4 RELEASE
1/7/2018 -- 07:24:40 - <Notice> - all 1 packet processing threads, 4 management
threads initialized, engine started.
```

Рисунок 3.5 – Запуск IDS Suricata.

Запустимо веб-сервер apache2 (рис. 3.6).

```
server@ubuntu:~$ sudo service apache2 start
[sudo] password for server:
* Starting web server apache2
*
server@ubuntu:~$
```

Рисунок 3.6 – Запуск веб-серверу apache2.

Використаємо утиліту hping3 для SYN-атаки на веб-сервер (рис. 3.7).

```
root@kali:~# sudo hping3 -c 10000 -d 120 -S -w 64 -p 80 --flood 192.168.1.130
HPING 192.168.1.130 (eth0 192.168.1.130): S set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
```

Рисунок 3.7 – Проведення SYN-атаки на веб-сервер.

Аналізуючи журнали подій IDS Suricata, ми помічаємо записи, які відповідають нашим налаштованим правилам (рис. 3.8). Зафіксовано наявність атаки типу DoS.

```
07/01/2018-07:32:03.208982  [**] [1:6:0] LOCAL DOS SYN packet flood outbound,
Potential DOS [**] [Classification: Misc activity] [Priority: 3] {TCP}
192.168.1.131:47241 -> 192.168.1.130:0
07/01/2018-07:32:07.843001  [**] [1:5:0] LOCAL DOS SYN packet flood inbound,
Potential DOS [**] [Classification: Misc activity] [Priority: 3] {TCP}
192.168.1.131:62261 -> 192.168.1.130:0
07/01/2018-07:32:07.843001  [**] [1:6:0] LOCAL DOS SYN packet flood outbound,
Potential DOS [**] [Classification: Misc activity] [Priority: 3] {TCP}
192.168.1.131:62261 -> 192.168.1.130:0
```

Рисунок 3.8 – Лог-файл після проведення DOS атаки.

Запустимо розроблений програмний модуль та виберемо лог-файл для аналізу, що лежить за адресою /var/log/suricata.log (рис. 3.9).

Open /var/log/suricata/fast.log Analyze

Af... dd.mm.y Be... dd.mm.y Conta... example: ddos Proto... Add ...

Sender IP: 255.255.255 Sender port: 8080 Receiver IP: 255.255.255 Receiver port: 8080

Delete rule Save to file

Commands for UFW

Рисунок 3.9 – Вибір лог-файлу для аналізу.

Встановимо правила для програмного модуля, де індикатор містить отримувачевий порт 80 та протокол TCP, оскільки атака була спрямована на веб-сервер за допомогою протоколу TCP. (рис. 3.10).

Рисунок 3.10 – Створення правила для модулю.

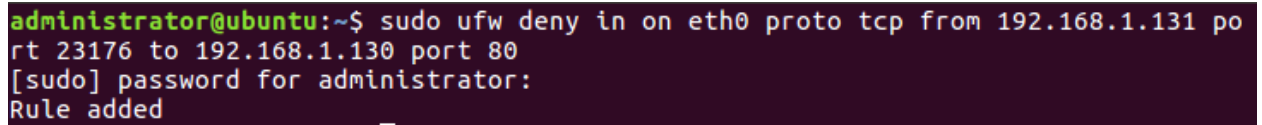
Аналізуючи лог-файл, програмний модуль згенерував команди для фаєрволу, використовуючи дані з індикаторів, які відповідають правилам модулю, і вивів їх у нижченаведений список (рис. 3.11).

```

sudo ufw deny in on eth0 proto tcp from 192.168.1.131 port 17776 to 192.168.1.130 port 80
sudo ufw deny in on eth0 proto tcp from 192.168.1.131 port 4534 to 192.168.1.130 port 80
sudo ufw deny in on eth0 proto tcp from 192.168.1.131 port 23176 to 192.168.1.130 port 80
sudo ufw deny in on eth0 proto tcp from 192.168.1.131 port 6944 to 192.168.1.130 port 80
sudo ufw deny in on eth0 proto tcp from 192.168.1.131 port 21079 to 192.168.1.130 port 80
sudo ufw deny in on eth0 proto tcp from 192.168.1.131 port 1930 to 192.168.1.130 port 80
sudo ufw deny in on eth0 proto tcp from 192.168.1.131 port 62965 to 192.168.1.130 port 80
  
```

Рисунок 3.11 – Сформовані команди для фаєрволу.

Скопіюємо будь-яку команду та запустимо її в терміналі (рис. 3.12).

A terminal window with a dark background. The prompt is 'administrator@ubuntu:~\$'. The command entered is 'sudo ufw deny in on eth0 proto tcp from 192.168.1.131 port 23176 to 192.168.1.130 port 80'. The output shows the password prompt '[sudo] password for administrator:' followed by 'Rule added' on a new line.

```
administrator@ubuntu:~$ sudo ufw deny in on eth0 proto tcp from 192.168.1.131 po
rt 23176 to 192.168.1.130 port 80
[sudo] password for administrator:
Rule added
```

Рисунок 3.12 – Запуск команди у терміналі.

Команда була успішно виконана, що призводить до ефективного блокування трафіку TCP, який походить з IP-адреси 192.168.1.131 та використовує порт 23176. Ці підозрілі з'єднання тепер будуть негайно зупинені та позбавлені доступу до системи, що допоможе попередити подальші атаки. будуть припинені та не матимуть доступу до системи.

ВИСНОВКИ

Дипломний проект зосереджений на розробці системи керування захистом серверів вебсайту від кібератак. Метою проекту є автоматизація процесу формування команд для актуатора IPS (Firewall) з метою виявлення та запобігання інформаційним атакам.

В процесі розробки була детально розглянута область інформаційної безпеки в комп'ютерних мережах, включаючи види мережових атак, шкідливе програмне забезпечення, системи виявлення вторгнень і сучасні АРТ-атаки.

Програмний модуль був реалізований на основі системи виявлення вторгнень Suricata за допомогою мови програмування Java 8.

Було досягнуто початкової мети, розглянуто загальні питання та на основі потреб був розроблений програмний модуль для керуванням подіями кібербезпеки власної мережі, серверів та сайтів.

Були проведені успішні тестування у віртуальній інформаційній системі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Закон України «Про захист інформації в інформаційно-телекомунікаційних системах».
2. Інформаційні системи і технології в управлінні / http://pidruchniki.com/74225/informatika/avtomatizovani_sistemi_upravlinnya_obroblennya_analizu_informatsiyi.
3. Аналіз сучасних систем виявлення вторгнень / О.В. Сєверінов, А.Г. Хренов, 2016р. 16 с.
4. Захищені операційні системи / Національний університет Львівська Політехніка, кафедра «БІКС» Лекція 3 2016р.
5. UFW / <http://integrator.adior.ru/index.php/linux-command/364-ufw-firewall-linux-pomogayushchij-upravlyat-iptables>.
6. Порівняння мов програмування / <http://lib.mdpu.org.ua/e-book/vstup/L4.html>
7. Мережеве обладнання – пристрої, з яких складається комп'ютерна мережа / <http://ittexnoall.com/index.php/osnovy-kompyutera/1002-merezheve-obladnannya-pristroji-z-yakikh-skladaetsya-komp-yuterna-merezha.html>
8. Характеристики мов програмування / Реферат «Можливості та призначення мови створення серверних сценаріїв» д.т.н. С.В. Ленков, 2015р. 3–4 с.

ДОДАТОК

```
package com.siem.module.indicator;

import java.io.Serializable;
import java.util.Date;

public class Indicator implements Serializable {

    private Date indicatorDate;

    private String protocol;

    private String message;

    private String senderIp;

    private String senderPort;

    private String receiverIp;

    private String receiverPort;

    public Indicator(Date indicatorDate, String protocol, String message,
String senderIp, String senderPort, String receiverIp, String receiverPort) {
        this.indicatorDate = indicatorDate;
        this.protocol = protocol;
        this.message = message;
        this.senderIp = senderIp;
        this.senderPort = senderPort;
        this.receiverIp = receiverIp;
        this.receiverPort = receiverPort;
    }

    public Date getIndicatorDate() {
        return indicatorDate;
    }

    public void setIndicatorDate(Date indicatorDate) {
        this.indicatorDate = indicatorDate;
    }

    public String getProtocol() {
        return protocol;
    }

    public void setProtocol(String protocol) {
        this.protocol = protocol;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public String getSenderIp() {
        return senderIp;
    }
```

```

    }

    public void setSenderId(String senderIp) {
        this.senderIp = senderIp;
    }

    public String getReceiverIp() {
        return receiverIp;
    }

    public String getSenderPort() {
        return senderPort;
    }

    public void setSenderPort(String senderPort) {
        this.senderPort = senderPort;
    }

    public void setReceiverIp(String receiverIp) {
        this.receiverIp = receiverIp;
    }

    public String getReceiverPort() {
        return receiverPort;
    }

    public void setReceiverPort(String receiverPort) {
        this.receiverPort = receiverPort;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Indicator)) return false;

        Indicator indicator = (Indicator) o;

        if (getProtocol() != null ?
!getProtocol().equals(indicator.getProtocol()) : indicator.getProtocol() !=
null)
            return false;
        if (getMessage() != null ?
!getMessage().equals(indicator.getMessage()) : indicator.getMessage() !=
null)
            return false;
        if (getSenderId() != null ?
!getSenderId().equals(indicator.getSenderId()) : indicator.getSenderId() !=
null)
            return false;
        if (getSenderPort() != null ?
!getSenderPort().equals(indicator.getSenderPort()) :
indicator.getSenderPort() != null)
            return false;
        if (getReceiverIp() != null ?
!getReceiverIp().equals(indicator.getReceiverIp()) :
indicator.getReceiverIp() != null)
            return false;
        return getReceiverPort() != null ?
getReceiverPort().equals(indicator.getReceiverPort()) :
indicator.getReceiverPort() == null;
    }

    @Override

```

```

    public int hashCode() {
        int result = getProtocol() != null ? getProtocol().hashCode() : 0;
        result = 31 * result + (getMessage() != null ?
getMessage().hashCode() : 0);
        result = 31 * result + (getSenderIp() != null ?
getSenderIp().hashCode() : 0);
        result = 31 * result + (getSenderPort() != null ?
getSenderPort().hashCode() : 0);
        result = 31 * result + (getReceiverIp() != null ?
getReceiverIp().hashCode() : 0);
        result = 31 * result + (getReceiverPort() != null ?
getReceiverPort().hashCode() : 0);
        return result;
    }

    @Override
    public String toString() {
        return "Indicator{" +
            "indicatorDate=" + indicatorDate +
            ", protocol='" + protocol + '\'' +
            ", message='" + message + '\'' +
            ", senderIp='" + senderIp + '\'' +
            ", senderPort='" + senderPort + '\'' +
            ", receiverIp='" + receiverIp + '\'' +
            ", receiverPort='" + receiverPort + '\'' +
            '}';
    }
}

```

```

package com.siem.module.indicator;

import com.siem.module.rule.Rule;

import java.util.HashSet;
import java.util.List;
import java.util.Set;
import java.util.function.Predicate;
import java.util.stream.Collectors;

public class IndicatorAnalyzer {

    public Set<Indicator> analyzeIndicators(Set<Indicator> indicators,
Set<Rule> ruleSet) {

        Set<Indicator> indicatorSet = new HashSet<>();

        ruleSet.forEach(rule -> {
            indicatorSet.addAll(indicators.stream().filter(new
Predicate<Indicator>() {
                @Override
                public boolean test(Indicator indicator) {
                    boolean isTrue = false;

                    if (rule.getAfter() != null) {
                        if
(indicator.getIndicatorDate().after(rule.getAfter())) {
                            isTrue = true;
                        }
                    }
                    if (rule.getBefore() != null) {
                        isTrue =
indicator.getIndicatorDate().before(rule.getBefore());
                    }

                    if(indicator.getMessage() != null && rule.getContains()
!= null && !rule.getContains().isEmpty()){
                        isTrue =
indicator.getMessage().toLowerCase().contains(rule.getContains().toLowerCase(
));
                    }
                    if(indicator.getProtocol() != null && rule.getProtocol()
!= null && !rule.getProtocol().isEmpty()){
                        isTrue =
indicator.getProtocol().equals(rule.getProtocol());
                    }
                    if(indicator.getSenderId() != null && rule.getSenderId()
!= null && !rule.getSenderId().isEmpty()){
                        isTrue =
indicator.getSenderId().equals(rule.getSenderId());
                    }
                    if(indicator.getSenderPort() != null &&
rule.getSenderPort() != null && !rule.getSenderPort().isEmpty()){
                        isTrue =
indicator.getSenderPort().equals(rule.getSenderPort());
                    }
                    if(indicator.getReceiverIp() != null &&
rule.getReceiverIp() != null && !rule.getReceiverIp().isEmpty()){
                        isTrue =

```

```

indicator.getReceiverIp().equals(rule.getReceiverIp());
    }
    if(indicator.getReceiverPort() != null &&
rule.getReceiverPort() != null && !rule.getReceiverPort().isEmpty()){
        isTrue =
indicator.getReceiverPort().equals(rule.getReceiverPort());
    }
    return isTrue;
    }
    }).collect(Collectors.toList());
});

    return indicatorSet;
}
}

```

```

package com.siem.module.indicator;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class IndicatorParser {

    private SimpleDateFormat simpleDateFormat = new
SimpleDateFormat("yyyy/MM/dd-HH:mm:ss.SSS");

    private Pattern protocolPattern = Pattern.compile("[{(A-Z)*}]");

    private Pattern ipPattern = Pattern.compile("[0-9.]+(:[0-9]*)");

    public Indicator parse(String indicatorData) throws ParseException {
        String[] indicatorDataArray = indicatorData.split("(.[*]+)");

        String date = indicatorDataArray[0].trim();
        String message = indicatorDataArray[1].trim();
        String connectionInfo = indicatorDataArray[2].trim();

        Date indicatorDate = this.simpleDateFormat.parse(date);

        String protocol = null;
        Matcher matcher = protocolPattern.matcher(connectionInfo);
        while (matcher.find()) {
            protocol = matcher.group(1);
        }

        ArrayList<String> ipAddressList = new ArrayList<>(2);
        ArrayList<String> ipPortsList = new ArrayList<>(2);
        matcher = ipPattern.matcher(connectionInfo);
        while (matcher.find()) {
            String ipAddress = matcher.group(1);
            if (!ipAddress.isEmpty() && ipAddress.length() > 4) {
                String ipInfo[] = ipAddress.split(":");
                ipAddressList.add(ipInfo[0]);
                ipPortsList.add(ipInfo[1]);
            }
        }

        Indicator p = null;
        if (ipAddressList.size() == 2 && !connectionInfo.isEmpty()) {
            p = new Indicator(indicatorDate, protocol.substring(1,
protocol.length() - 1), message, ipAddressList.get(0), ipPortsList.get(0),
ipAddressList.get(1), ipPortsList.get(1));
        }

        return p;
    }
}

```



```

package com.siem.module.rule;

import java.io.Serializable;
import java.util.Date;

public class Rule implements Serializable {

    private Date after;

    private Date before;

    private String contains;

    private String protocol;

    private String senderIp;

    private String senderPort;

    private String receiverIp;

    private String receiverPort;

    public Rule() {
    }

    public Rule(Date after, Date before, String contains, String protocol,
String senderIp, String senderPort, String receiverIp, String receiverPort) {
        this.after = after;
        this.before = before;
        this.contains = contains;
        this.protocol = protocol;
        this.senderIp = senderIp;
        this.senderPort = senderPort;
        this.receiverIp = receiverIp;
        this.receiverPort = receiverPort;
    }

    public Date getAfter() {
        return after;
    }

    public void setAfter(Date after) {
        this.after = after;
    }

    public Date getBefore() {
        return before;
    }

    public void setBefore(Date before) {
        this.before = before;
    }

    public String getContains() {
        return contains;
    }

    public void setContains(String contains) {
        this.contains = contains;
    }

    public String getProtocol() {

```

```

        return protocol;
    }

    public void setProtocol(String protocol) {
        this.protocol = protocol;
    }

    public String getSenderId() {
        return senderIp;
    }

    public void setSenderId(String senderIp) {
        this.senderIp = senderIp;
    }

    public String getSenderPort() {
        return senderPort;
    }

    public void setSenderPort(String senderPort) {
        this.senderPort = senderPort;
    }

    public String getReceiverIp() {
        return receiverIp;
    }

    public void setReceiverIp(String receiverIp) {
        this.receiverIp = receiverIp;
    }

    public String getReceiverPort() {
        return receiverPort;
    }

    public void setReceiverPort(String receiverPort) {
        this.receiverPort = receiverPort;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Rule)) return false;

        Rule rule = (Rule) o;

        if (getAfter() != null ? !getAfter().equals(rule.getAfter()) :
rule.getAfter() != null) return false;
        if (getBefore() != null ? !getBefore().equals(rule.getBefore()) :
rule.getBefore() != null) return false;
        if (getContains() != null ? !getContains().equals(rule.getContains())
: rule.getContains() != null)
            return false;
        if (getProtocol() != null ? !getProtocol().equals(rule.getProtocol())
: rule.getProtocol() != null)
            return false;
        if (getSenderId() != null ? !getSenderId().equals(rule.getSenderId())
: rule.getSenderId() != null)
            return false;
        if (getSenderPort() != null ?
!getSenderPort().equals(rule.getSenderPort()) : rule.getSenderPort() != null)
            return false;
        if (getReceiverIp() != null ?

```

```

!getReceiverIp().equals(rule.getReceiverIp()) : rule.getReceiverIp() != null)
    return false;
    return getReceiverPort() != null ?
getReceiverPort().equals(rule.getReceiverPort()) : rule.getReceiverPort() ==
null;
}

@Override
public int hashCode() {
    int result = getAfter() != null ? getAfter().hashCode() : 0;
    result = 31 * result + (getBefore() != null ? getBefore().hashCode()
: 0);
    result = 31 * result + (getContains() != null ?
getContains().hashCode() : 0);
    result = 31 * result + (getProtocol() != null ?
getProtocol().hashCode() : 0);
    result = 31 * result + (getSenderId() != null ?
getSenderId().hashCode() : 0);
    result = 31 * result + (getSenderPort() != null ?
getSenderPort().hashCode() : 0);
    result = 31 * result + (getReceiverIp() != null ?
getReceiverIp().hashCode() : 0);
    result = 31 * result + (getReceiverPort() != null ?
getReceiverPort().hashCode() : 0);
    return result;
}

@Override
public String toString() {
    String rule = "";
    if(this.getAfter() != null){
        rule += "after " + this.getAfter() + " ";
    }
    if(this.getBefore() != null){
        rule += "before " + this.getBefore() + " ";
    }
    if(this.getContains() != null && !this.getContains().isEmpty()){
        rule += "contains " + this.getContains() + " ";
    }
    if(this.getProtocol() != null && !this.getProtocol().isEmpty()){
        rule += "protocol " + this.getProtocol() + " ";
    }
    if(this.getSenderId() != null && !this.getSenderId().isEmpty()){
        rule += "sender ip " + this.getSenderId() + " ";
    }
    if(this.getSenderPort() != null && !this.getSenderPort().isEmpty()){
        rule += "sender port " + this.getSenderPort() + " ";
    }
    if(this.getReceiverIp() != null && !this.getReceiverIp().isEmpty()){
        rule += "receiver ip " + this.getReceiverIp() + " ";
    }
    if(this.getReceiverPort() != null &&
!this.getReceiverPort().isEmpty()){
        rule += "receiver port " + this.getReceiverPort() + " ";
    }

    return rule.trim().toUpperCase();
}
}

```

```

package com.siem.module.rule;

import java.io.*;
import java.net.URL;
import java.util.HashSet;
import java.util.Set;

public class RulesManager {

    private File rulesFile;

    private Set<Rule> ruleSet = new HashSet<>();

    public RulesManager() {
        this.rulesFile = new File("module.rules");
        if (!this.rulesFile.exists()) {
            try {
                this.rulesFile.createNewFile();
            } catch (IOException e) {
                e.printStackTrace();
            }
        } else {
            if (this.rulesFile.length() > 0) {
                FileInputStream fileInputStream = null;
                try {
                    fileInputStream = new
FileInputStream(this.rulesFile.getAbsolutePath());
                    ObjectInputStream objectInputStream = new
ObjectInputStream(fileInputStream);
                    this.ruleSet = (Set<Rule>)
objectInputStream.readObject();
                    objectInputStream.close();
                } catch (FileNotFoundException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                } catch (ClassNotFoundException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    public void saveRules() {
        if (this.rulesFile.exists()) {
            this.rulesFile.delete();
            try {
                this.rulesFile.createNewFile();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        FileOutputStream fileOutputStream = null;
        ObjectOutputStream objectOutputStream = null;
        try {
            fileOutputStream = new FileOutputStream(this.rulesFile);
            objectOutputStream = new ObjectOutputStream(fileOutputStream);
            objectOutputStream.writeObject(this.ruleSet);
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {

```

```

        outputStream.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

}

public void addRule(Rule rule) {
    this.ruleSet.add(rule);
}

public File getRulesFile() {
    return rulesFile;
}

public void setRulesFile(File rulesFile) {
    this.rulesFile = rulesFile;
}

public Set<Rule> getRuleSet() {
    return ruleSet;
}

public void setRuleSet(Set<Rule> ruleSet) {
    this.ruleSet = ruleSet;
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof RulesManager)) return false;

    RulesManager that = (RulesManager) o;

    if (getRulesFile() != null ?
!getRulesFile().equals(that.getRulesFile()) : that.getRulesFile() != null)
        return false;
    return getRuleSet() != null ? getRuleSet().equals(that.getRuleSet())
: that.getRuleSet() == null;
}

@Override
public int hashCode() {
    int result = getRulesFile() != null ? getRulesFile().hashCode() : 0;
    result = 31 * result + (getRuleSet() != null ?
getRuleSet().hashCode() : 0);
    return result;
}
}

```

```
package com.siem.module.rule;

public class UfwRuleFactory {

    private String UFW_RULE_PATTERN = "sudo ufw %s in on %s proto %s from %s
port %s to %s port %s";

    public String generateRule(String action, String interfaceName, String
proto, String senderIp, String senderPort, String receiverIp, String
receiverPort){
        return String.format(UFW_RULE_PATTERN, action, interfaceName,
proto.toLowerCase(), senderIp, senderPort, receiverIp, receiverPort);
    }

}
```

```

package com.siem.module;

import com.siem.module.indicator.Indicator;
import com.siem.module.indicator.IndicatorParser;
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;

import java.io.IOException;
import java.text.ParseException;

public class Main extends Application {

    private Stage primaryStage;

    private AnchorPane rootLayout;

    public static void main(String[] args) {
        launch(args);
    }

    public void initRootLayout() {
        try {
            FXMLLoader loader = new FXMLLoader();

loader.setLocation(Main.class.getClassLoader().getResource("view/main_view.fxml"));

            rootLayout = (AnchorPane) loader.load();
            Scene scene = new Scene(rootLayout);
            primaryStage.setScene(scene);
            primaryStage.show();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void start(Stage primaryStage) {
        this.primaryStage = primaryStage;
        this.primaryStage.setTitle("SIEM module");
        this.initRootLayout();
    }
}

```

```

package com.siem.module;

import com.siem.module.indicator.Indicator;
import com.siem.module.indicator.IndicatorAnalyzer;
import com.siem.module.indicator.IndicatorParser;
import com.siem.module.rule.Rule;
import com.siem.module.rule.RulesManager;
import com.siem.module.rule.UfwRuleFactory;
import javafx.collections.FXCollections;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.Initializable;
import javafx.scene.control.*;
import javafx.scene.input.Clipboard;
import javafx.scene.input.ClipboardContent;
import javafx.scene.input.MouseButton;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.stage.FileChooser;

import java.io.File;
import java.io.IOException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.HashSet;
import java.util.ResourceBundle;
import java.util.Set;
import java.util.function.Predicate;
import java.util.stream.Stream;

public class MainViewController implements Initializable {

    private Set<Indicator> indicatorsSet;

    private File logFile;
    private RulesManager rulesManager;
    private IndicatorParser indicatorParser;
    private IndicatorAnalyzer indicatorAnalyzer;
    private UfwRuleFactory ufwRuleFactory;

    private SimpleDateFormat simpleDateFormat = new
SimpleDateFormat("yyyy.MM.dd");

    public AnchorPane anchor_pane;

    public Button button_choose_file;
    public Button button_delete_rule;
    public Button button_save_rules;
    public Button button_analyze;
    public Button button_add_rule;

    public Label label_file_path;

    public ChoiceBox choice_box_protocol;

    public TextField field_date_after;
    public TextField field_date_before;
    public TextField field_contains;
    public TextField field_sender_ip;

```

```

public TextField field_sender_port;
public TextField field_receiver_ip;
public TextField field_receiver_port;

public ListView list_rules;
public ListView list_commands;

private ContextMenu contextMenu;
private MenuItem itemMenuCopy;

@Override
public void initialize(URL location, ResourceBundle resources) {
    this.rulesManager = new RulesManager();
    this.indicatorsSet = new HashSet<>();
    this.indicatorParser = new IndicatorParser();
    this.indicatorAnalyzer = new IndicatorAnalyzer();
    this.ufwRuleFactory = new UfwRuleFactory();

    this.choice_box_protocol.setItems(FXCollections.observableArrayList("TCP",
    "UDP", "ICMP", "IP"));
    this.list_rules.getItems().addAll(this.rulesManager.getRuleSet());
    this.itemMenuCopy = new MenuItem("Copy");
    this.contextMenu = new ContextMenu();
    this.contextMenu.getItems().add(itemMenuCopy);
    this.itemMenuCopy.setOnAction(actionEvent -> {
        final Clipboard clipboard = Clipboard.getSystemClipboard();
        final ClipboardContent content = new ClipboardContent();

        content.putString(this.list_commands.getSelectionModel().getSelectedItem().toString());
        clipboard.setContent(content);
    });
    this.list_commands.setOnMouseClicked(event -> {
        if(event.getButton().equals(MouseButton.SECONDARY)) {
            this.contextMenu.show(this.list_commands, event.getScreenX(),
            event.getScreenY());
        } else {
            this.contextMenu.hide();
        }
    });
}

public void openLogFile(ActionEvent actionEvent) {
    FileChooser fileChooser = new FileChooser();
    fileChooser.setTitle("Open log file");
    this.logFile =
fileChooser.showOpenDialog(this.label_file_path.getScene().getWindow());
    if(logFile != null){
        this.label_file_path.setText(this.logFile.getAbsolutePath());
    }
}

public void analyzeLog(ActionEvent actionEvent) {
    this.indicatorsSet.clear();
    try (Stream<String> stream =
Files.lines(Paths.get(this.logFile.getAbsolutePath()))) {
        stream.forEach(s -> {
            try {
                indicatorsSet.add(this.indicatorParser.parse(s));
            } catch (ParseException e) {
                e.printStackTrace();
            }
        });
}

```

```

    } catch (IOException e) {
        e.printStackTrace();
    }

    this.list_commands.getItems().clear();
    Set<Indicator> indicatorsForUfw =
this.indicatorAnalyzer.analyzeIndicators(this.indicatorsSet,
this.rulesManager.getRuleSet());
    indicatorsForUfw.forEach(indicator -> {
        String ufwCommand = ufwRuleFactory.generateRule("deny", "eth0",
indicator.getProtocol(), indicator.getSenderIp(), indicator.getSenderPort(),
indicator.getReceiverIp(), indicator.getReceiverPort());
        this.list_commands.getItems().add(ufwCommand);
    });

}

public void addRule(ActionEvent actionEvent) {
    Date before = null;
    Date after = null;
    try {
        if(!this.field_date_before.getText().isEmpty()){
            before =
this.simpleDateFormat.parse(this.field_date_before.getText());
        }
        if(!this.field_date_after.getText().isEmpty()){
            after =
this.simpleDateFormat.parse(this.field_date_after.getText());
        }
    } catch (ParseException e) {
        e.printStackTrace();
    }
    String contains = this.field_contains.getText();
    Object protocolObject =
this.choice_box_protocol.getSelectionModel().getSelectedItem();
    String protocol = "";
    if(protocolObject != null){
        protocol = protocolObject.toString();
    }
    String senderIp = this.field_sender_ip.getText();
    String senderPort = this.field_sender_port.getText();
    String receiverIp = this.field_receiver_ip.getText();
    String receiverPort = this.field_receiver_port.getText();
    Rule rule = new Rule(before, after, contains, protocol, senderIp,
senderPort, receiverIp, receiverPort);
    this.rulesManager.addRule(rule);
    this.list_rules.getItems().add(rule.toString());
}

public void deleteRule(ActionEvent actionEvent) {
    Object ruleObject =
this.list_rules.getSelectionModel().getSelectedItem();
    this.list_rules.getItems().remove(ruleObject);
    this.rulesManager.getRuleSet().removeIf(new Predicate<Rule>() {
        @Override
        public boolean test(Rule rule) {
            return rule.toString().equals(ruleObject.toString());
        }
    });
}

public void saveRules(ActionEvent actionEvent) {
    this.rulesManager.saveRules();
}
}

```

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.ChoiceBox?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.ListView?>
<?import javafx.scene.control.Separator?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.control.ToolBar?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>

<AnchorPane xmlns="http://javafx.com/javafx/8.0.121"
xmlns:fx="http://javafx.com/fxml/1" fx:id="anchor_pane"
fx:controller="com.siem.module.MainViewController">
    <children>
        <ToolBar prefHeight="40.0" prefWidth="200.0"
AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0">
            <items>
                <Button fx:id="button_choose_file" mnemonicParsing="false"
onAction="#openLogFile" text="Open" />
                <Label fx:id="label_file_path" prefWidth="250.0"
styleClass="file_label" stylesheets="@../styles.css" text="Please select log
file">
                    <font>
                        <Font size="14.0" />
                    </font>
                    <padding>
                        <Insets bottom="2.0" left="5.0" right="5.0" top="1.0"
/>
                    </padding>
                </Label>
                <Button fx:id="button_analyze" mnemonicParsing="false"
onAction="#analyzeLog" text="Analyze" />
            </items>
        </ToolBar>
        <HBox alignment="CENTER_LEFT" prefHeight="40.0" style="-fx-background-
color: #ffffcc;" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0"
AnchorPane.topAnchor="40.0">
            <children>
                <Label text="After:">
                    <HBox.margin>
                        <Insets />
                    </HBox.margin>
                    <padding>
                        <Insets left="10.0" right="10.0" />
                    </padding>
                </Label>
                <TextField fx:id="field_date_after" prefWidth="80.0"
promptText="dd.mm.yyyy" />
                <Label text="Before">
                    <HBox.margin>
                        <Insets left="10.0" right="10.0" />
                    </HBox.margin>
                </Label>
                <TextField fx:id="field_date_before" prefWidth="80.0"
promptText="dd.mm.yyyy" />
                <Label text="Contains:">
                    <HBox.margin>
                        <Insets left="10.0" right="10.0" />

```

```

        </HBox.margin></Label>
        <TextField fx:id="field_contains" promptText="example: ddos" />
        <Label text="Protocol:">
            <HBox.margin>
                <Insets left="10.0" right="10.0" />
            </HBox.margin>
        </Label>
        <ChoiceBox fx:id="choice_box_protocol" prefWidth="100.0" />
        <Button fx:id="button_add_rule" mnemonicParsing="false"
onAction="#addRule" text="Add rule">
            <HBox.margin>
                <Insets left="10.0" right="10.0" />
            </HBox.margin>
        </Button>
    </children>
    <padding>
        <Insets bottom="10.0" top="10.0" />
    </padding>
</HBox>
<HBox alignment="CENTER_LEFT" prefHeight="40.0" style="-fx-background-
color: #ffffcc;" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0"
AnchorPane.topAnchor="85.0">
    <children>
        <Label text="Sender IP:">
            <HBox.margin>
                <Insets left="10.0" right="10.0" />
            </HBox.margin>
        </Label>
        <TextField fx:id="field_sender_ip" prefWidth="100.0"
promptText="255.255.255.255" />
        <Label text="Sender port:">
            <padding>
                <Insets left="10.0" right="10.0" />
            </padding>
        </Label>
        <TextField fx:id="field_sender_port" prefWidth="60.0"
promptText="8080" />
        <Label text="Reciever IP">
            <HBox.margin>
                <Insets left="10.0" right="10.0" />
            </HBox.margin>
        </Label>
        <TextField fx:id="field_receiver_ip" layoutX="83.0"
layoutY="18.0" prefWidth="100.0" promptText="255.255.255.255" />
        <Label layoutX="183.0" layoutY="22.0" text="Receiver port:">
            <padding>
                <Insets left="10.0" right="10.0" />
            </padding>
        </Label>
        <TextField fx:id="field_receiver_port" layoutX="269.0"
layoutY="18.0" prefWidth="60.0" promptText="8080" />
    </children>
</HBox>
<ListView fx:id="list_rules" layoutX="14.0" layoutY="125.0"
prefHeight="150.0" prefWidth="200.0" AnchorPane.leftAnchor="0.0"
AnchorPane.rightAnchor="0.0" />
<HBox alignment="CENTER_RIGHT" layoutY="275.0" prefHeight="40.0"
style="-fx-background-color: #ffffcc;" AnchorPane.leftAnchor="0.0"
AnchorPane.rightAnchor="0.0">
    <children>
        <Button fx:id="button_delete_rule" mnemonicParsing="false"
onAction="#deleteRule" text="Delete rule">
            <HBox.margin>

```

```

        <Insets right="15.0" />
    </HBox.margin>
</Button>
<Button fx:id="button_save_rules" mnemonicParsing="false"
onAction="#saveRules" text="Save to file">
    <HBox.margin>
        <Insets right="15.0" />
    </HBox.margin>
</Button>
</children>
</HBox>
<Separator AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0"
AnchorPane.topAnchor="315.0" />
<VBox layoutY="315.0" prefHeight="200.0" prefWidth="100.0"
AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0">
    <children>
        <Label text="Commands for UFW">
            <padding>
                <Insets bottom="10.0" left="10.0" right="10.0" top="10.0"
/>
            </padding>
        </Label>
        <ListView fx:id="list_commands" prefHeight="200.0"
prefWidth="200.0" />
    </children>
</VBox>
</children>
<padding>
    <Insets bottom="10.0" />
</padding>
</AnchorPane>

```

ВІДОМІСТЬ ДИПЛОМНОЇ РОБОТИ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	A4		Титульний лист		
2	A4		Завдання на дипломну роботу		
3	A4		Пояснювальна записка		
4	A4		Додатки		
5	A4		Відгук		
6	A4		Рецензія		
7	A4		Відомість		