МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Навчально-науковий інститут новітніх освітніх технологій
Кафедра комп'ютерної інженерії

Козловський Олег Васильович

Система детекції вторгнень на основі Mikrotik / Creating an IDS protection system based on Mikrotik

спеціальність: 123 – Комп'ютерна інженерія
освітньо-професійна програма – Комп'ютерна інженерія

Кваліфікаційна

Виконав: студент групи КІз-41
Козловський Олег Васильович

Науковий Керівник
к.т.н. Мельник Г.М.

ТЕРНОПІЛЛЬ-2023

РЕЗЮМЕ

Кваліфікаційна робота на тему «Система детекції вторгнень на основі Mikrotik» зі спеціальності 123 «Комп'ютерна інженерія» освітнього ступеня «бакалавр» містить 62 сторінки пояснюючої записки, 11 рисунків, 8 таблиць, 3 додатки. Обсяг графічного матеріалу 2 аркуші формату А3.

Метою кваліфікаційної роботи є розроблення системи детекції вторгнень на основі обладнання Mikrotik.

Методи дослідження включають методи фізичної і логічної структуризації комп'ютерних мереж, методи структурного програмування, теорія графів, елементи математичної логіки.

Розроблено алгоритми керування правилами детекції атак, які враховують специфіку мережі і потенційні загрози. Створено об'єктну модель ПЗ, структуру правила в JSON форматі. Створено структуру мережі з системою IDS, включаючи налаштування маршрутизаторів, комутаторів та інших мережевих пристроїв. В якості компонетів взято системи Suricata, Elasticsearch, веб-інтерфейс Scirius. Систему IDS встановлено на ОС Linux Ubuntu. Розроблено UML діаграму розгортання системи.

Ключові слова: СИСТЕМА ДЕТЕКЦІЇ ВТОРГНЕНЬ, ФІЛЬТРАЦІЯ ТРАФІКУ, МАРШРУТИЗАТОР.

RESUME

Qualification thesis “Creating an IDS protection system based on Mikrotik” in the specialty 123 "Computer Engineering" of bachelor education degree contains 62 pages of explanatory note, 11 figures, 8 tables, 3 appendices. The amount of graphic material is 2 sheets of A3 format.

The purpose of the qualification work is to develop an intrusion detection system based on Mikrotik equipment.

Research methods include methods of physical and logical structuring of computer networks, methods of structural programming, graph theory, elements of mathematical logic.

Algorithms for controlling attack detection rules that take into account the specifics of the network and potential threats have been developed. An object model of the software and a rule structure in JSON format were created. A network structure with an IDS system was created, including the configuration of routers, switches, and other network devices. Suricata, Elasticsearch, and the Scirius web interface were used as components. The IDS system is installed on the Linux Ubuntu OS. A UML diagram of the system deployment is developed.

Keywords: INTRUSION DETECTION SYSTEM, TRAFFIC FILTERING, ROUTER.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

DMZ	–	Demilitarized Zone
IDS	–	Intrusion Detection System
NIDS	–	network-based IDS
HIDS	–	host-based IDS
VLAN	–	Virtual Local Area Network
VPN	–	Virtual Private Network

ВСТУП

Актуальність розробки і впровадження систем детекції вторгнень (IDS) полягає в наступних аспектах.

Захист від зловмисних атак: Інформаційні системи і мережі постійно піддаються різноманітним видам кібератак, таким як вторгнення, шкідливе програмне забезпечення, порт-сканування, DDoS-атаки тощо. Система IDS може ефективно виявляти такі загрози та надавати захист шляхом реагування на них.

Раннє виявлення загроз: IDS дозволяє виявляти аномальну активність і неправильну поведінку в мережі ще до того, як вона стане серйозною загрозою для безпеки. Це дозволяє оперативно реагувати на потенційні загрози та запобігати втратам даних і порушенням безпеки.

Зменшення ризиків та витрат: IDS допомагає зменшити ризики і потенційні втрати, пов'язані з кібератаками, знизити витрати на відновлення, а також запобігти можливим правовим наслідкам.

Законодавчі вимоги: Багато галузей, таких як фінансові установи, медичні організації та установи, що працюють з особистими даними, зобов'язані виконувати вимоги щодо захисту даних та конфіденційності. Розробка IDS допоможе виконувати ці вимоги та забезпечити високий рівень безпеки.

Практична цінність рішень на основі Mikrotik та відкритого ПЗ моніторингу/детекції для полягає в наступному.

Розроблюване рішення допомагає виявляти потенційні вторгнення і атаки на мережу і оперативно реагувати на них. Це допомагає зберегти конфіденційність, цілісність та доступність мережевих ресурсів.

Розроблюване рішення Mikrotik забезпечує мінімальні витрати на обладнання. Система детекції атак може бути легко інтегрована з іншими системами безпеки. Це дозволяє створити комплексну систему безпеки мережі зі збалансованим підходом до виявлення, захисту та реагування на загрози.

Правила детекції вторгнень для IDS є логічними виразами, які описують характеристики атаки або небезпеки, яку система спробує виявити. Зокрема, в системі Suricata, структура правила складається з різних полів, які визначають параметри атаки, такі як джерело, призначення, протокол, порт і т.д. Правило також містить умови, що визначають, які події або активності в мережі можуть вказувати на вторгнення. Важливі елементи правила включають ключові слова, модифікатори, опції, маски та інші параметри, які специфікують деталі детекції атаки. Правила детекції вторгнень є ключовим інструментом для виявлення потенційних загроз у мережевому трафіку та забезпечення безпеки інформаційних систем. Правила та сигнатури дозволяють системі активно моніторити трафік і сповіщати про можливі вторгнення.

Метою кваліфікаційної роботи є розроблення системи детекції вторгнень на основі обладнання Mikrotik. Для досягнення мети потрібно виконати такі завдання:

– проаналізувати функції та структуру систем детекції вторгнень;

– проаналізувати специфіку роботи маршрутизаторів Mikrotik;

– розробити алгоритми керування правилами детекції атак;

– розробити структуру мережі із системою IDS;

– програмно реалізувати розроблені алгоритми;

– здійснити тестування системи детекції вторгень.

Об'єктом розробки є система детекції вторгнень на основі обладнання Mikrotik, а предметом розробки є функції, структура та алгоритми керування правилами детекції атак, а також структура мережі із системою IDS.

Практичні цінність роботи полягає в тому, що розробка системи на основі обладнання Mikrotik дозволить ефективно виявляти потенційні загрози та забезпечувати захист мережевої інфраструктури.

За результатами роботи опубліковано тези доповіді на VII науково-практичній конференції «Інтелектуальні комп’ютерні системи та мережі» [1]. Копії публікації наведено у додатку В.

1 ОГЛЯД СИСТЕМ ДЕТЕКЦІЇ ВТОРГНЕНЬ

1.1 Технічні заходи безпеки мережі

Технічні (логічні) заходи безпеки – це програмні засоби, використовувані для обмеження доступу суб'єктів до об'єктів. Це можуть бути компоненти операційних систем, окремі пакети безпеки, додатки, апаратні мережні пристрої, протоколи, механізми шифрування, матриці контролю доступу. Ці заходи безпеки працюють на різних рівнях у мережі або системах, але при цьому повинна бути забезпечена їхня спільна робота для захисту від несанкціонованого доступу до ресурсів і гарантій доступності, цілісності й конфіденційності ресурсів. Технічні заходи захищають цілісність і доступність ресурсів, обмежуючи число суб'єктів, які можуть мати до них доступ, а також конфіденційність ресурсів, запобігаючи їх розкриття неавторизованим суб'єктам [2-3].

Архітектура мережі може бути побудована й реалізовано за допомогою декількох логічних засобів захисту, що забезпечують ізоляцію й захист оточення. Мережа може бути ізольована фізично (стінами, виділеними приміщеннями), або логічно (окремими адресними просторами, підмережами, сегментами, керованими комунікаційними потоками між сегментами). Часто дуже важливо контролювати взаємодію різних сегментів між собою. На рисунку 1.1 показаний приклад того, як компанія може сегментувати свої мережі й організувати взаємодію між сегментами. У наведеному прикладі компанія не прагне, щоб між внутрішньою мережею й демілітаризованою зоною (DMZ) були відкриті й необмежені комунікаційні маршрути. Звичайно внутрішнім користувачам немає необхідності прямо звертатися до систем в DMZ, а виключення таких маршрутів взаємодії знижує можливості для внутрішніх атак на ці системи. Крім того, якщо атака з Інтернету успішно компрометує систему в DMZ, атакуючий не повинен одержати при цьому можливість простого доступу у внутрішню мережу, для чого повинен застосовуватися відповідний тип логічної ізоляції.

На цьому прикладі також показано, що керуючий сегмент може взаємодіяти з усіма іншими мережними сегментами, але ці мережні сегменти не можуть взаємодіяти з керуючим сегментом, тому що в ньому перебувають консолі, що управляють міжмережевими екранами й IDS, і немає причин для взаємодії з ними користувачів і інших адміністраторів [4,5].

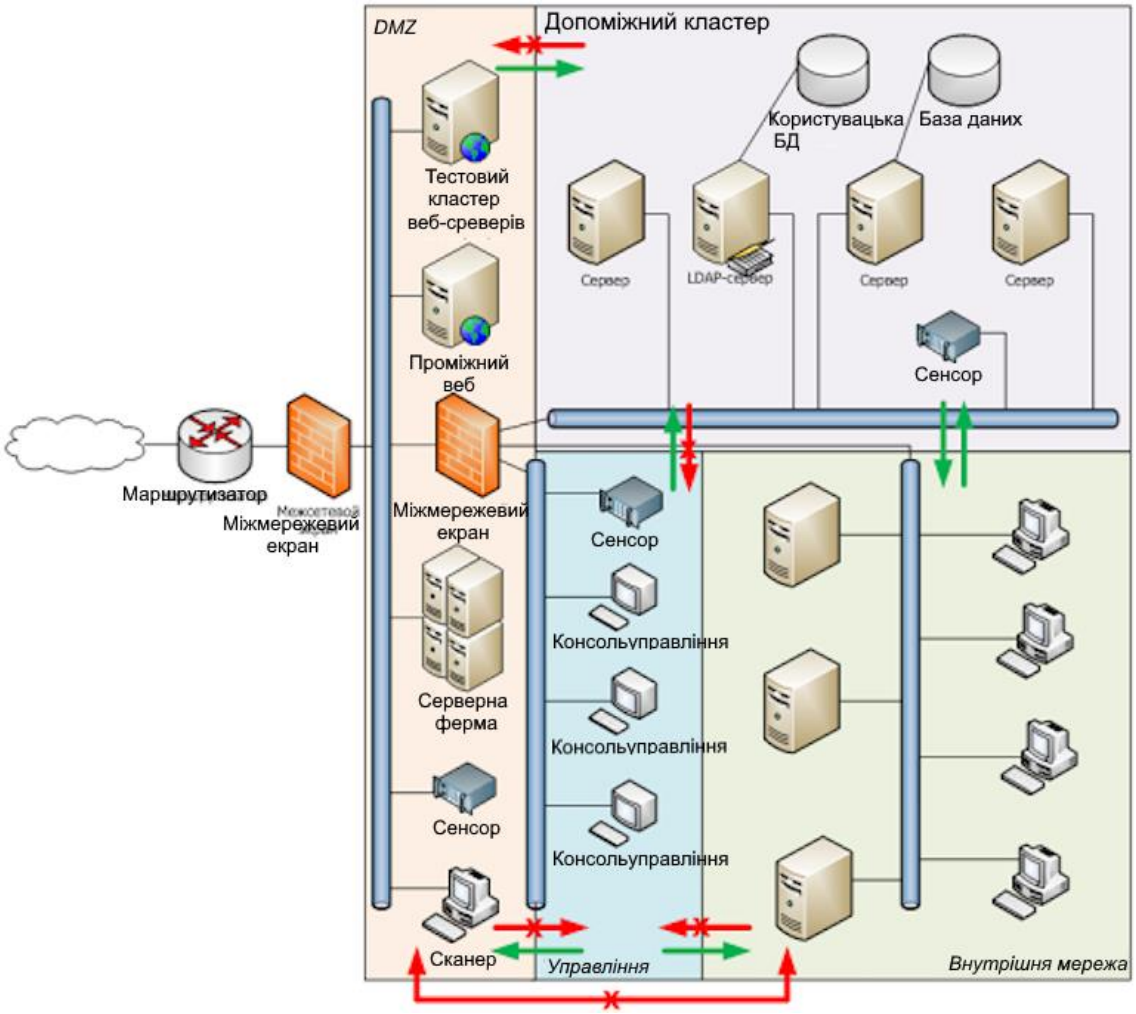

Рисунок 1.1- Сегментація мережі на технічному рівні управляє порядком взаємодії різних сегментів мережі

1.2 Системи виявлення вторгень

Моніторинг керування доступом – це метод відстеження тих, хто намагається одержати доступ до певних ресурсів компанії. Це важливий

детективний механізм, для реалізації якого існують різні технології, такі як IDS, IPS, мережні сніфери, хости-приманки (honeypot). Недостатньо просто вкласти гроші в антивірус і міжмережевий екран, компаніям необхідний контроль своїх власних внутрішніх мереж.

Системи виявлення вторгнень (IDS – intrusion detection system) відрізняються від традиційних міжмережевих екранів, тому що вони створені для виявлення недоліків у системі безпеки – несанкціонованого використання або атак комп'ютерів, мереж або телекомунікаційної інфраструктури. IDS дозволяють знизити збитки від хакерських атак, злому критичних комп'ютерів і мережних пристроїв. Основне завдання IDS – відзначати підозрілі дії в мережі й вчасно повідомляти про них адміністратора (за допомогою передачі повідомлення на консоль керування, відправлення SMS-Повідомлення на мобільний телефон і т.п.), або навіть автоматично вносити зміни в настроювання ACL міжмережевого екрана. Засоби IDS можуть переглядати потоки даних, знаходячи в них послідовності бітів, які можуть свідчити про сумнівні дії або події, або здійснювати моніторинг системних журналів і інших файлів журналювання діяльності. Слід виявляти будь-яку ненормальну поведінку, яка може свідчити про вторгнення (або спробі вторгнення) [5-9].

Хоча існують різні різновиди IDS, усі вони мають три загальні компоненти: сенсори, аналізатори й адміністративні інтерфейси. Сенсори збирають трафік або дані про дії користувачів і відправляють їхнім аналізаторам, які шукають у них підозрілі дії. У випадку виявлення аналізатором таких дій (на які він запрограмований), він відправляє відповідні повідомлення в адміністративний інтерфейс. Існує два основні типи IDS:

-рівня мережі (network-based) відслідковують увесь мережний трафік,

-рівня вузла (host-based) аналізують дії в рамках однієї комп'ютерної системи.

IDS можуть бути настроєні для виявлення атак, аналізу журналів аудита, переривання з'єднань, сповіщення адміністратора про поточні атаки, захисту системних файлів, вказівки на уразливості, які повинні бути враховані, а також для допомоги у відстеженні дій окремих хакерів.

IDS рівня мережі (NIDS – network-based IDS) використовують сенсори, що є окремими комп'ютерами із установленим на них спеціальним програмним забезпеченням, або спеціальними виділеними пристроями (appliance). Кожний сенсор має мережну карту (NIC – network interface card), що працює в режимі прослуховування мережі (promiscuous mode). Звичайні мережні карти одержують тільки мережні пакети, адресовані цій мережній карті, широкомовні (broadcast) і багатоадресні (multicast) пакети. Драйвер мережної карти копіює дані з передавального середовища й відправляє й відправляє його стеку мережних протоколів для обробки. Якщо мережна карта перебуває в режимі прослуховування, драйвер мережної карти захоплює весь трафік, робить копії всіх пакетів, а потім передає одну копію в стек TCP, а другу копію – аналізатору, для пошуку певних шаблонів (сигнатур).

NIDS контролює мережний трафік й не може побачити дії, котрі відбуваються усередині окремого комп'ютера. Для відстеження таких дій слід використовувати IDS рівня вузла.

IDS рівня вузла (HIDS – host-based IDS) може бути встановлена на окремі робочі станції й/або сервери для виявлення небажаних або аномальних дій. HIDS звичайно використовуються для забезпечення впевненості, що користувачі не видаляють системні файли, не змінюють важливі настроювання або піддають систему ризику іншими способами. Так, якщо NIDS розуміє й контролює мережний трафік, засоби HIDS обмежуються тільки самим комп'ютером. HIDS не розуміє й не відслідковує мережний трафік, а NIDS не контролює дії усередині системи. Кожен із цих засобів має свої завдання й виконує їх своїми способами [2, 10-13].

У більшості середовищ системи HIDS установлюються тільки на критичні сервери, а не на кожний комп'ютер у мережі, оскільки це могло б викликати істотне підвищення навантаження на комп'ютери й на адміністраторів. Системи HIDS і NIDS можуть бути одного з наступних типів:

1. Сигнатурні (signature based):
   - відслідковуючі шаблони (pattern matching);
   - відслідковуючі стан (stateful matching).

2. Засновані на аномаліях (anomaly based):

- засновані на статистичних аномаліях (statistical anomaly-based);
- засновані на аномаліях протоколів (protocol anomaly-based);
- засновані на аномаліях трафіка (traffic anomaly-based).

3. Засновані на правилах (rule-based) або евристичні (heuristic-based).

Детекція атак на основі списків сигнатур.

Знання про окремі атаки накопичуються виробниками IDS і зберігаються у вигляді моделей, що відбивають процес їх виконання. Ці моделі називаються сигнатурами. Як тільки виявляється новий вид атаки, виробник IDS створює відповідну сигнатуру, яка надалі використовується при перевірці мережного трафіка для виявлення такої ж атаки. Дозволяються будь-які дії, які не були ідентифіковані як атака.

Прикладом сигнатури є мережний пакет, у якому адреса відправника й одержувача збігаються – це, так звана, Land-атака. В Land-Атаці хакер змінює заголовок пакета й коли система одержувача відправляє відправникові відповідь, вона відправляє його на свою ж адресу. Зараз це виглядає досить безпечно, але усе ще є вразливі системи, що не мають програмного коду, що дозволяє розпізнати таку ситуацію, яка приводить до них «зависанню» або перезавантаженню.

Сигнатурні IDS є найбільш популярними IDS у наш час, але їх ефективність прямо залежить від регулярності відновлення баз сигнатур, як в антивірусному програмному забезпеченні. Цей тип IDS практично не захищає від нових атак, тому що він не може їх розпізнати до появи їх сигнатур у його базі даних.

Розглянемо IDS на основі станів і саме поняття стану системи. Кожна зміна в роботі системи (вхід користувача, запуск додатка, взаємодія додатків, введення даних і т.д.) приводить до зміни стану. З технічної точки зору, усі операційні системи й додатки являють собою просту безліч рядків команд, написаних для виконання певних функцій над даними. Команди працюють зі змінними, які містять дані. Коли додатки взаємодіють один з одним, вони

заповнюють порожні змінні спеціальними наборами команд. Таким чином, перехід стану – це коли міняється значення змінної, що відбувається постійно в будь-якій системі.

При проведенні атак, відбуваються відповідні зміни станів (дії). Стан – це «знімок» (snapshot) значень операційної системи в оперативній, напівпостійній і постійної областях пам'яті. В IDS на основі станів первісним станом є стан перед початком атаки, а скомпрометованим станом – стан після успішного вторгнення. IDS має правила, які описують послідовність переходів стану, що свідчать про атаку, що відбувається. Дії, які відбуваються між первісним і скомпрометованим станами – це саме те, що шукає даний тип IDS. Він відправляє сигнал небезпеки, якщо будь-яка послідовність переходів стану збігається з попередньо настроєними правилами. Цей тип IDS здійснює пошуки сигнатур атак у контексті потоку дій, а не просто дивиться окремі пакети. Він також може виявити тільки відомі атаки й вимагає частого відновлення своїх сигнатур.

IDS на основі статистичних аномалій (statistical anomaly–based IDS) – це системи, засновані на поведінці. Вони не використовують сигнатури. Замість цього вони створюють профіль «нормальної» діяльності, працюючи в режимі навчання. Цей профіль будується на основах постійного аналізу, що відбувається в середовищі діяльності. Точність профілю і якість захисту залежить від часу знаходження IDS у режимі навчання. Після створення профілю, увесь наступний трафік і діяльність рівняються з ним. Усі що не схоже на профіль, уважається атакою, про яку направляється відповідне повідомлення. Такі IDS використовують складні статистичні алгоритми, вишукуючи аномалії в мережному трафіку й діях користувачів. Кожному пакету надається рейтинг його «аномальності», який указує на ступінь його відхилення від нормального профілю. Якщо рейтинг вище певного порога «нормального» поведінки, виконується заздалегідь визначена дія.

Перевагою IDS на основі статистичних аномалій є їхня можливість реагувати на нові типи атак, у тому числі атаки «нульвого дня», для яких ще немає сигнатур або заплаток (патчів). Ці IDS також можуть виявляти «низькі й

повільні» атаки, при яких атакуючий намагається залишитися нижче порога спрацьовування засобів моніторингу, відправляючи пакети потроху протягом тривалого періоду часу. IDS на основі статистичних аномалій можуть виявляти ці типи атак, оскільки вони досить відрізняються від профілю.

Вкрай складно створити для мережі якісний профіль «нормальної» діяльності, який не приводить до величезної кількості неправильних спрацьовувань. Це пов'язане з тим, що в мережі відбуваються постійні зміни. Часто це приводить до того, що компанії просто відключають свої IDS, через те, що вони вимагають украй багато часу для своєї належної підтримки. Щоб скоротити число неправильних спрацьовувань, потрібний дуже висококваліфікований IT-Персонал. Також, важливим моментом є правильне визначення порога спрацьовування IDS.

Якщо атакуючий виявляє наявність IDS у мережі, він спробує визначити її тип, щоб спробувати обійти її. Для поведінкової IDS атакуючий може спробувати сполучити свою діяльність із шаблоном поведінки мережного трафіка. Якщо це вдасться йому, його діяльність буде виглядати нормальною для IDS і тому залишиться не виявленою. Крім того, надто важливо гарантувати відсутність атак під час роботи IDS у режимі навчання, тому що інакше IDS буде вважати такі атаки «нормальною» діяльністю й не буде повідомляти про них у майбутньому.

Якщо компанія вирішує використовувати IDS на основі статистичних аномалій, вона повинна переконатися, що її співробітники, які будуть впроваджувати й підтримувати систему, розбираються в проведенні аналізу протоколів і пакетів. Це пов'язане з тим, що такі IDS (на відміну від IDS інших типів) відправляють малоінформативні повідомлення й мережний інженер повинен у кожному випадку розбиратися, у чому насправді полягає проблема. Наприклад, сигнатурна IDS повідомляє тип виявленої атаки, IDS на основі правил повідомляє яке правило було порушено. IDS на основі статистичних аномалій просто повідомляють про те, що відбулося щось «ненормальне», не відповідне до профілю.

При підготовці до атаки, хакери найчастіше першим кроком визначають, чи установлений IDS у мережі, яку вони збираються атакувати. Якщо IDS установлений, вони проводять Dos-атаку на нього, щоб порушити його роботу. Іншою тактикою є відправлення IDS некоректних даних, які змусять IDS відправити повідомлення про почату атаку, хоча ніякої атаки в дійсності не буде. Це робиться для того, щоб добитися відключення IDS фахівцями компанії через її «неправильну» роботу, або щоб відволікти увагу цих фахівців на аналіз некоректних пакетів, поки буде відбуватися реальна атака.

Сигнатурні IDS також відомі як системи виявлення некоректного використання (misuse-detection system), а поведінкові IDS – як системи, засновані на профілях (profile-based system).

Встановлення правильних порогів статистично значимих відхилень є критичним фактором для ефективного використання поведінкової IDS. Якщо поріг встановлено занадто низько, це може призводити до частого помилкового виявлення звичайної активності як атаки (false positive, "помилкове спрацювання"). З іншого боку, якщо поріг встановлено занадто високо, деякі шкідливі дії можуть залишитись непоміченими (false negative, "нерозпізнавання"). Отже, правильне налаштування порогів є вирішальним для досягнення балансу між точністю виявлення та зменшенням помилкових спрацювань IDS.

Коли IDS виявляє атаку, вона може робити різні дії, залежно від своїх можливостей і настроєних на ній політик. IDS може відправити повідомлення на консоль керувань, щоб повідомити про атаку відповідним до фахівців; відправити повідомлення по електронній пошті або на мобільний телефон фахівцеві, відповідальному за реакцію на атаки; скинути з'єднання, з якого зареєстрована атака; або перенастроїти маршрутизатор або міжмережевий екран, щоб спробувати зупинити всі наступні схожі атаки. Реакція може варіюватися від блокування конкретної IP-Адреси, до перенаправлення або блокування окремих видів діяльності.

IDS на основі статистичних аномалій можуть використовувати фільтри, засновані на аномаліях протоколів. Такі IDS мають знання про кожний

протокол, який вони контролюють. Аномалії протоколів виявляються на підставі формату й поведінки протоколу. IDS будує модель (профіль) «нормального» використання кожного протоколу. Потрібно мати на увазі, що теоретичне використання протоколів описане у відповідних RFC, але їх реальна робота майже завжди відрізняється від теоретичної, оскільки виробники програмного забезпечення в більшості випадків не строго випливають RFC. Таким чином, більшість профілів окремих протоколів є сумішшю з офіційних і реальних варіантів їх використання. Коли IDS включається в роботу, вона шукає аномалії, які не збігаються із профілями, побудованими для конкретних протоколів. Хоча в самих операційних системах і додатках досить експлуатованих вразливостей, більшість успішних атак використовують уразливості самих протоколів.

IDS на основі аналізу аномалій трафіку. Більшість поведінкових IDS мають фільтри, засновані на аномаліях трафіків, які виявляють зміни в шаблонах трафіка, наприклад DOS-Атаки або поява нових сервісів у мережі. Створений профіль є якимось базисом звичайного трафіку середовища, і весь наступний трафік порівнюється із цим профілем. Як і у всіх фільтрах, тут потрібна настроювання порога спрацьовування для зменшення числі неправильних спрацьовувань (як неправильних дозволів, так і хибних заборон). Такий тип IDS також здатний виявляти невідомі атаки.

IDS на основі правил (rule-based IDS) використовують інший підхід, що відрізняється від підходу сигнатурних IDS або IDS на основі статистичних аномалій. Наприклад, якщо сигнатурна IDS виявляє пакет, у якому всі прапори заголовка TCP установлені в «1», він знає, що це свідчить про xmas-атаці, і відправляє відповідне оповіщення. IDS на основі статистичних аномалій також достатньо прямолінійні. Наприклад, якщо Боб зареєструвався на своєму комп'ютері в 6 ранку, а відповідно до його профілю це ненормально, IDS відправляє повідомлення про це, оскільки це виглядає як дії, які повинні бути проаналізовані. IDS на основі правил діє хитріше залежно від складності правил, що застосовуються.

1.3 Системи аналізу на основі правил

IDS на основі правил схожі на експертні системи. Експертна система заснована на базі знань (knowledge base), механізмі логічних виводів (inference engine) і програмуванні на основі правил (rule-based programming). Знання являють собою правила, а аналізовані дані – факти. Знання системи описуються за допомогою програмування на основах правил

(ЯКЩО *ситуація* ТОДІ *дія*).

Ці правила застосовуються до фактів, до даних, одержуваним сенсором, або до контрольованих систем. Наприклад, у сценарії 1 IDS одержує дані з журналу аудита системи й тимчасово зберігає їхній базі даних фактів, як показано на малюнку 1.2. Потім до цих даних застосовуються попередньо настроєні правила, які перевіряють, чи їсти що-небудь підозріле в що відбувся подіях. У сценарії правило вказує

```
«ЯКЩО користувач admin створив File11 И створив File12
ПРИ ЦЬОМУ обидва ці файлу перебувають в директорії
ПОТІМ запустив «Управлінську програму1»
ТОДІ відправити повідомленняХ».
```

Це правило визначає, що якщо користувач root створив два файли в одній директорії, а потім запустив певну адміністративну утиліту, повинне бути відправлене повідомлення.

Механізм логічних виводів реалізує якийсь штучний інтелект у даному процесі. Він може робити висновки, створюючи нову інформацію з отриманих даних, використовуючи правила.

Таким чином, в IDS заснованих на правилах, що працюють аналогічно експертним системам, IDS збирає дані із сенсорів або з журналів аудита, а механізм логічних виводів обробляє їх, використовуючи попередньо

запрограмовані правила. Якщо характеристики задовольняють правилу, відправляється відповідне повідомлення або виконується певна дія для розв'язку виниклої проблеми.

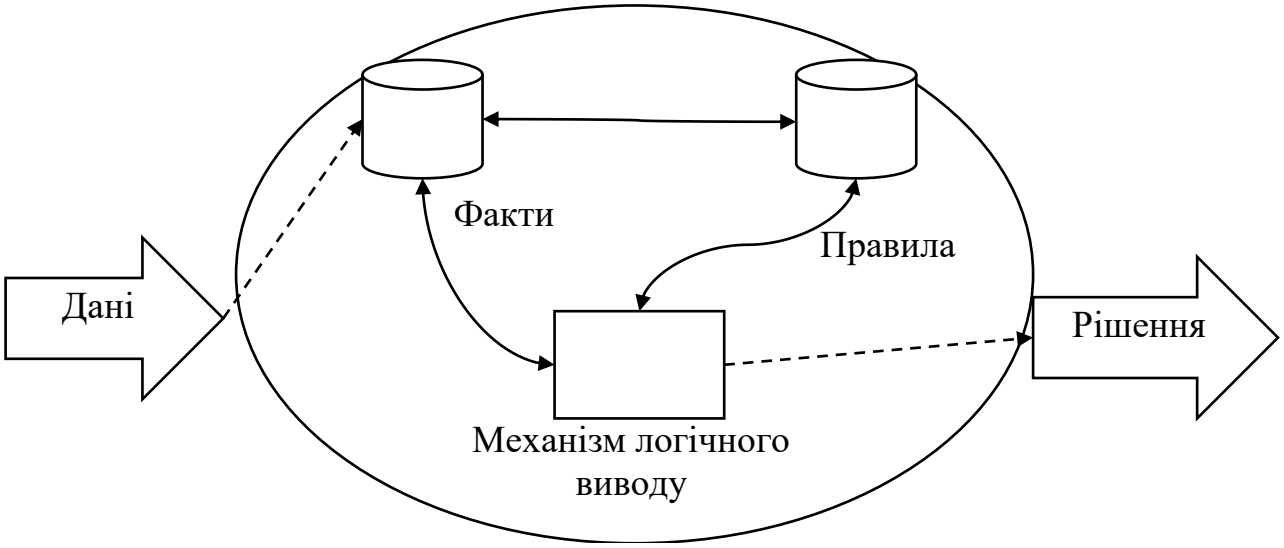

Рисунок 1.2 - Компоненти IDS на правилах і експертної системи

1.4 Постановка задач кваліфікаційної роботи

Важливо розуміти характеристики, що відрізняють різні типи технологій IDS. Підсумуємо різницю**:**

1. сигнатурні (signature), відслідковуючі шаблони (pattern matching).

- Сигнатури повинні постійно поновлятися, не можуть виявити нові атаки

Існують Два типи:

1. відслідковують шаблони (pattern matching) – порівнюють пакети із сигнатурами;

2. відслідковують стан (stateful matching) – порівнюють дії із шаблонами.

- засновані на аномаліях (anomaly-based);

- засновані на поведінці системи (behavioral-based), які вивчають «нормальну» діяльність у середовищі.

Вони можуть виявляти нові атаки, Також називаються поведінковими або евристичними. Три різних підтипи типи:

1. Засновані на статистичних аномаліях (statistical anomaly-based) – створюють профіль «нормальної» діяльності й порівнюють реальну діяльність із цим профілем
2. Засновані на аномаліях протоколів (protocol anomaly-based) – виявляють факти незвичайного використань протоколів
3. Засновані на аномаліях трафіка (traffic anomaly-based) – виявляють незвичайні дії в мережному трафіках

Системи засновані на правилах (rule-based)

а. Використовують програмування на правилах ЯКЩО/ТОДІ у рамках експертних систем.

б. Використовують експертну систему, що має характеристики штучного інтелекту.

в. Більш складні правила проте не завжди можуть виявляти нові атаки.

В корпоративних мережах України використання маршрутизаторів Mikrotik є поширеним, особливо в малих і середніх підприємствах, тому розробка системи на основі цієї платформи дозволяє широкому колу користувачів впровадити ефективні засоби детекції вторгнень. Пристрої Mikrotik пропонуть потужні можливості налаштування мережі, включаючи функції маршрутизації, фаєрволу та VPN, що дозволяє інтегрувати систему детекції вторгнень в комплексний захист мережі. Розробка системи на основі Mikrotik забезпечує зручний та простий інтерфейс управління, що сприяє простоті використання та адміністрування системи детекції вторгнень.

Отже актуальним є розроблення системи детекції вторгнень на основі пристроїв MikroTik, яка буде виявляти та сповіщати про потенційні вторгнення та зловмисну активність у мережі. Метою кваліфікаційної роботи є розроблення системи детекції вторгнень на основі обладнання Mikrotik.

При розробленні ПЗ потрібно навести ООП специфікації класів які

визначають поведінку та властивості об'єктів. Специфікації описує публічні методи, поля, властивості та інші елементи, які визначають, як можна взаємодіяти з об'єктами класу. Специфікація класу може також включати інформацію про виключення та консистентність стану об'єктів після виконання методів.

Метою кваліфікаційної роботи є розроблення системи детекції вторгнень на основі обладнання Mikrotik. Для досягнення мети потрібно виконати такі завдання:

– проаналізувати функції та структуру систем детекції вторгнень;

– проаналізувати специфіку роботи маршрутизаторів Mikrotik;

– розробити алгоритми керування правилами детекції атак;

– розробити структуру мережі із системою IDS;

– програмно реалізувати розроблені алгоритми;

– здійснити тестування системи детекції вторгень.

2 РОЗРОБЛЕННЯ СТРУКТУРИ І КОМПОНЕНТ СИСТЕМИ ДЕТЕКЦІЇ ВТОРГНЕНЬ

2.1 Структура мережі з маршрутизатором Mikrotik

Сенсори систем детекції

IDS мережного рівня використовують сенсори для здійснення моніторингу. Сенсор, який працює як аналітичний двигун (analysis engine), розміщається в контрольованому мережному сегменті. Сенсор одержує «сирі» (raw) дані від генератора подій, як показано на рисунку 2.1, і порівнює їх з базою даних сигнатур, профілем або моделлю. Якщо виявляється якийсь збіг, який свідчить про підозрілу активність, сенсор працює як модуль реагування для визначення виду дій, які потрібно почати (повідомлення через систему миттєвих повідомлень, мобільний телефон, електронну пошту, перенастроювання міжмережевого екрана і т.д.). Сенсор призначений для фільтрації отриманих ним даних, відкидаючи непотрібну інформацію й виявляючи підозрілу діяльність.

Системі IDS складніше працювати в комутованому середовищі порівняно з традиційними, некомутованими середовищами, оскільки дані в ньому передаються через незалежні віртуальні канали, а не транслюються. Тому в комутованому середовищі сенсор має бути під'єднаний до спеціального порту (spanning port) на комунікаційному пристрої, на який автоматично копіюється (віддзеркалюється в термінології Mikrotik) весь трафік, що проходить через усі віртуальні канали. Це дасть змогу сенсору мати доступ до всього трафіку, що проходить через комутовану мережу.

Консоль моніторингу контролює всі сенсори й надає мережному персоналу огляд роботи всіх сенсорів у рамках усієї мережі. Розміщення сенсорів є одним із критичних етапів конфігурування IDS. Компанія може розмістити один сенсор перед міжмережевим екраном для виявлення атак, а інший сенсор за міжмережевим екраном (у мережному периметрі) для виявлення реальних атак.

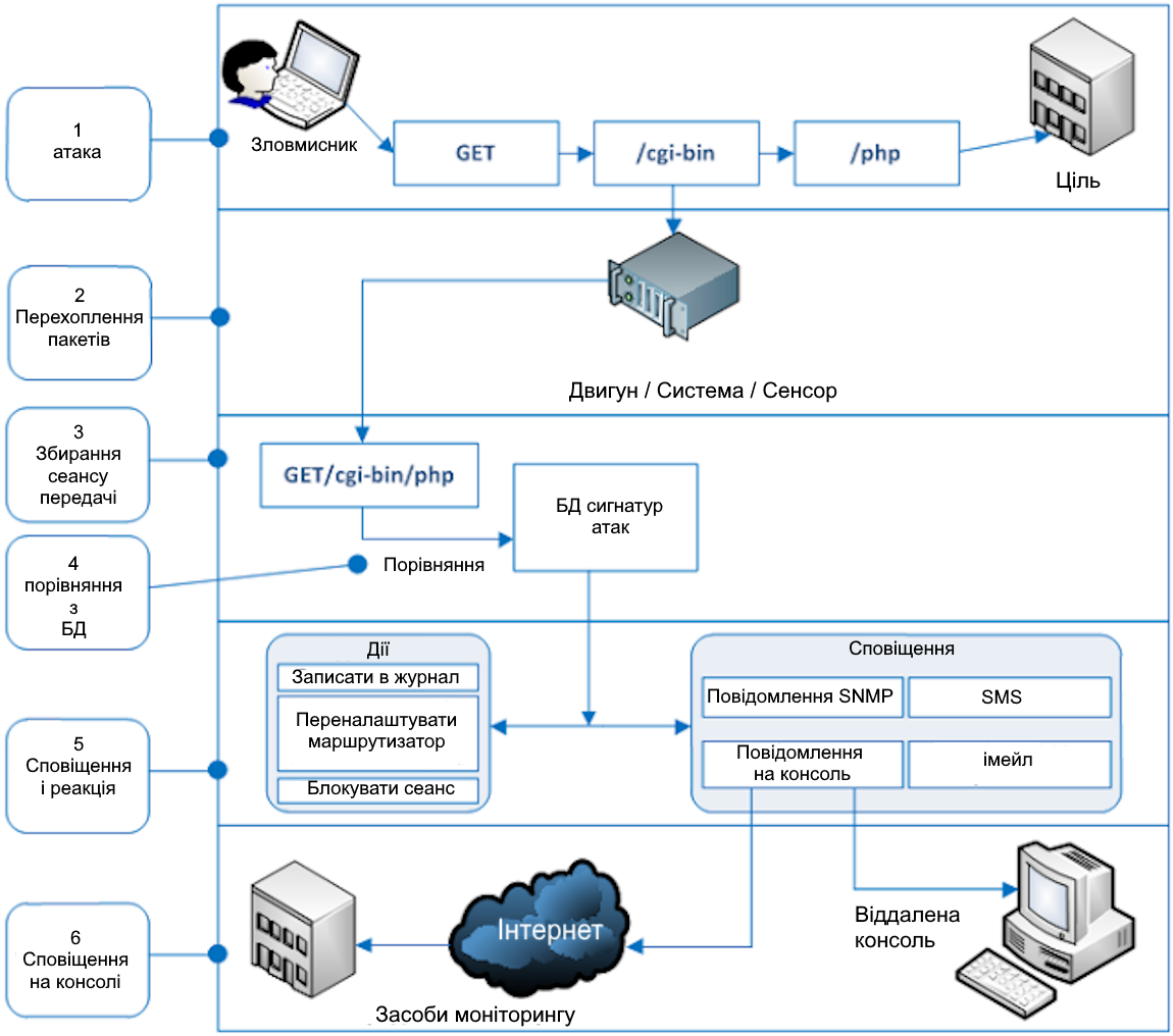

Рисунок 2.1 - Основна архітектура IDS

Сенсори слід також розміщати у висококритичних областях, DMZ і в екстрамережах. Рисунок 2.2 показує сенсори, що передають дані на центральну консоль.

IDS може бути централізованою (наприклад, вбудована в міжмережевий екран), або розподіленою (з безліччю сенсорів, розподілених по всій мережі).

Якщо обсяг мережного трафіку перевищує поріг IDS, окремі атаки можуть залишитися непоміченими. Кожна IDS має власний поріг, про який обов'язково потрібно знати до покупки й впровадження IDS.

У середовищах з дуже великим обсягом трафіка слід розміщати безліч сенсорів, щоб забезпечити впевненість, що всі пакети проаналізовані. Якщо необхідно оптимізувати пропускну здатність і швидкість роботи мережі, можна настроїти різні сенсори на аналіз кожного пакета на відповідність різним

(окремим) наборам сигнатур. Таким чином, навантаження по аналізі пакетів може бути розбита на кілька різних точок.

Рисунок 2.2 - Сенсори в кожному сегменті мережі

Функція Port mirroring в RouterOS є механізмом, який дозволяє копіювати (дзеркалювати) мережевий трафік, який проходить через певний порт мережевого пристрою, на інший порт. Цей механізм дозволяє аналізувати або моніторити мережевий трафік для налагодження, безпеки або інших цілей без втручання в нормальний шлях передачі трафіку.

Коли функція Port mirroring ввімкнена для певного порту, всі пакети, які надходять або виходять через цей порт, будуть копіюватися і перенаправлятися на інший порт, який визначений як «слухаючий» порт або «порт дзеркалення». На цьому слухаючому порту можна налаштувати засоби моніторингу або аналізу, такі як мережеві аналізатори, IDS (інтрузійні системи виявлення), аналізатори пакетів і т.д., для отримання інформації про трафік.

Функція Port mirroring в RouterOS дозволяє операторам мережі здійснювати моніторинг і аналіз мережевого трафіку в реальному часі без необхідності встановлення додаткових пристроїв або втручання в нормальну роботу мережі. Це дозволяє виявляти аномалії, атаки або інші проблеми з безпекою та продуктивністю мережі, що є важливим для забезпечення стабільності та ефективності мережевої інфраструктури.

2.2 Система виявлення вторгнень Suricata

Suricata – це система виявлення вторгнень (IDS) та система запобігання вторгненням (IPS), яка аналізує мережевий трафік і виявляє потенційні загрози на основі заданих правил.

Правила фільтрації Suricata описують певні відомі сигнатури атак або патерни поведінки, які можуть вказувати на наявність загрози. Вони включають різноманітні параметри, такі як тип атаки, підписи пакетів, значення полів заголовків пакетів тощо.

Ці правила фільтрації можуть бути використані для спостереження за мережевим трафіком і виявлення потенційних атак, включаючи відомі вразливості програмного забезпечення, шкідливі програми, аномальні поведінки тощо. Правила фільтрації оновлюються та розширюються з часом для виявлення нових загроз і атак.

Suricata може використовуватися як самостійний IDS/IPS, або як компонент більшої системи безпеки мережі для забезпечення виявлення та запобігання вторгненням.

Правило фільтрації в IDS Suricata

Правило фільтрації в IDS Suricata складається з декількох частин, які визначаються певними ключовими словами та параметрами [9]. Основні частини правила включають:

1. Характеристики трафіку: Визначаються типи трафіку, з якими правило працює, наприклад, TCP або UDP, порти джерела та призначення і т.д.

2. Повідомлення (msg): Текстове повідомлення, яке вказує на призначення правила або описує виявлену атаку.

3. Умови виявлення (content, byte_test, byte_jump): Ці частини визначають шаблон байтів або послідовностей байтів, які IDS шукає в мережевому трафіку. Вони можуть містити значення байтів, довжини, зміщення, умови перевірки байтів тощо.

4. Метаданія (metadata): Додаткова інформація про правило, така як рівень серйозності (severity), класифікація (classtype), ідентифікатор (sid), версія (rev) тощо.

Це лише загальна структура правила, і її можна додатково налаштовувати залежно від потреб. Наприклад, можна вказати додаткові параметри для фільтрації пакетів за допомогою опцій, таких як часові межі (time), напрямок потоку (flow) та інші.

Приклад сигнатури.

Один з прикладів відомої сигнатури атаки, яку можна використовувати в IDS Suricata, - це сигнатура атаки «ET EXPLOIT Possible SSLv3 Outbound Message». Ця сигнатура спрямована на виявлення використання застарілого протоколу SSLv3, який має відомі вразливості. Приклад правила фільтрації для цієї атаки в може виглядати так:

```
1 alert
2 tcp any any -> any any
3 (msg:»ET EXPLOIT Possible SSLv3 Outbound Message»;
  flow:established, to_server;
  content:»|16 03|»; depth:2;
  byte_jump:4,2,relative,little,bitstring,from_beginning,dec;
  isdataat:!1,relative;
  byte_test:1,>,127,0,relative; content:»|03 00|»;
  distance:0; within:2;
  byte_jump:2,2,relative,little,bitstring,from_beginning,dec;
  isdataat:!1,relative; byte_test:1,>,127,0,relative;
  content:»|01|»; distance:0; within:1;
  byte_jump:1,2,relative,little,bitstring,from_beginning,dec;
  isdataat:!1,relative; byte_test:1,>,127,0,relative;
  content:»|03|»; distance:0; within:1;
  byte_jump:1,2,relative,little,bitstring,from_beginning,dec;
  isdataat:!1,relative; byte_test:1,>,127,0,relative;
  metadata:severity «High»; classtype:attempted-recon;
  sid:123456789; rev:1
```

Це правило шукає певний шаблон байтів у TCP-пакетах, які вказують на використання SSLv3 протоколу. Якщо цей шаблон виявляється в мережевому трафіку, IDS Suricata генерує сповіщення (alert) або виконує певні дії відповідно до налаштувань. Список сигнатур атак постійно оновлюється, нові

сигнатури у відкритих базах сигнатур атак, таких як Emerging Threats [7] и Snort Subscriber Rule Set [23].

У прикладі цифрам позначено складові правила 1 – дія Action, 2 – Header, 3- Rule option [9, 8]. Допустимими діями є

– alert – згенерувати попередження;

– pass – припинити подальшу перевірку пакета;

– drop – відкинути пакет і згенерувати попередження;

– reject – надіслати відправнику відповідного пакета повідомлення про помилку RST/ICMP недосяжності;

– rejectsrc – те саме, що й просто reject;

– rejectdst – надіслати пакет з помилкою RST/ICMP одержувачу відповідного пакета;

– rejectboth – надіслати пакети помилок RST/ICMP обом сторонам розмови.

Параметр Protocol в підписі вказує, якого протоколу він стосується. Можна вибрати один з чотирьох основних протоколів: TCP (для tcp-трафіку), UDP, ICMP, IP (замінює «all» або «any»). Можна вибрати один протокол 7-го прикладного рівня, зокрема HTTP, FTP, SMB, DNS, DCERPC, NFS та ін.

2.3 Алгоритми керування правилами детекції атак

2.3.1 Розробимо алгоритми керування правилами фільтрації та сповіщення про атаки. Спочатку розробимо UML діаграми класів та Use Case UML діаграми для алгоритму, який запитує в користувача параметри правила фільтрації Suricata.

Клас UserInterface представляє інтерфейс користувача, через який користувач може вводити параметри правила фільтрації.

Клас FilterRule представляє правило фільтрації, яке складається з параметрів, таких як умови, дія, пріоритет, напрямок, джерело та призначення.

Клас SuricataFilteringAlgorithm представляє алгоритм обробки правил фільтрації в Suricata. Включає методи для взаємодії з інтерфейсом користувача та створення та застосування правил фільтрації.

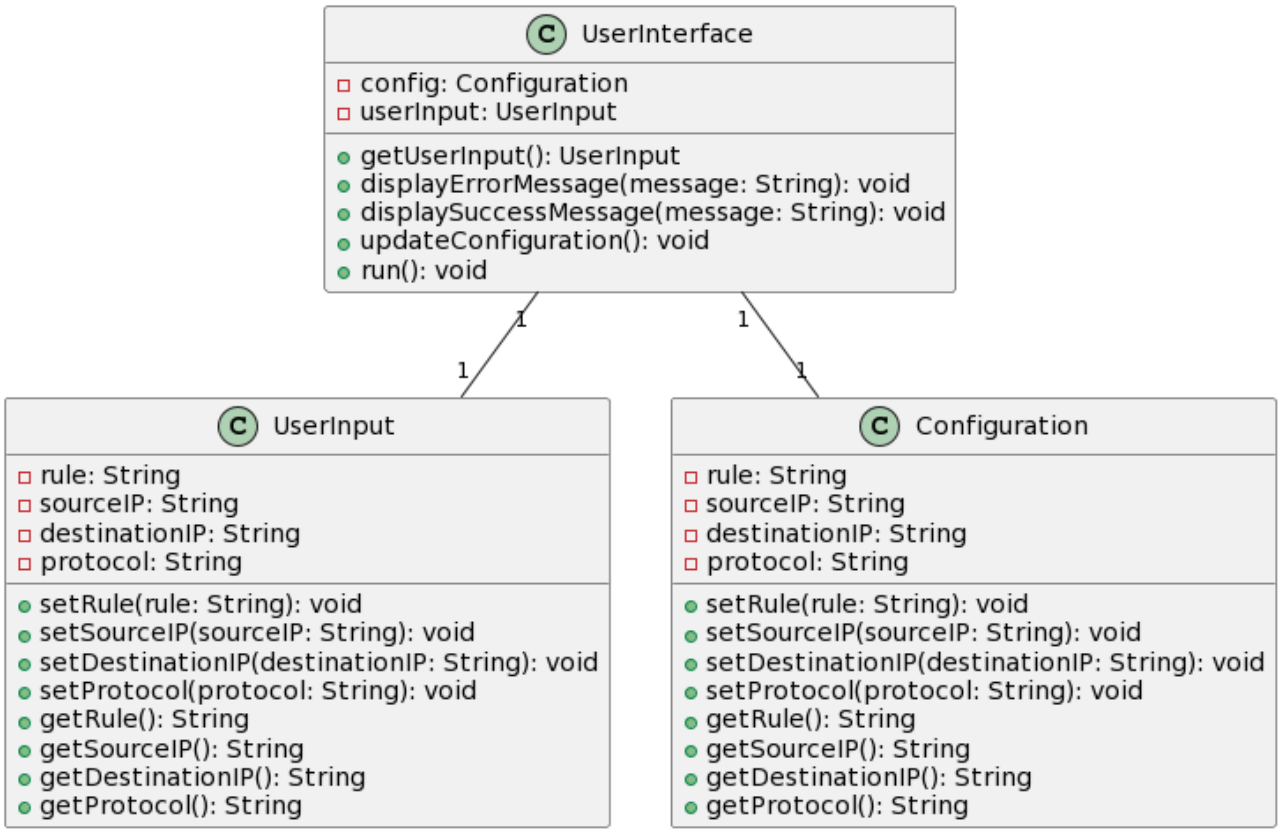

Рисунок 2.3 - Загальна діаграма класів

Таблиця 2.1 - Методи класу UserInterface

Метод	Специфікація
getUserInput (): UserInput	○ Опис: Отримує введення користувача з інтерфейсу. ○ Повертає: Об'єкт UserInput, що представляє введення користувача
displayErrorMessage (message: String): void	○ Опис: Відображає повідомлення про помилку на інтерфейсі. ○ Параметри: ▪ message: String - Повідомлення про помилку, яке потрібно відобразити.
displaySuccessMessage (message: String): void	○ Опис: Відображає повідомлення про успіх на інтерфейсі. ○ Параметри: ▪ message: String - Повідомлення про успіх, яке потрібно відобразити.

Продовження таблиця 2.1 - Методи класу UserInterface

updateConfiguration(): void	○ Опис: Оновлює конфігурацію на основі введення користувача. ○ Примітка: Цей метод використовує властивості userInput та config для оновлення конфігурації.
run(): void	○ Опис: Виконує роботу інтерфейсу користувача. ○ Примітка: Цей метод є головною точкою входу в інтерфейс користувача. Він ініціалізує необхідні компоненти, отримує введення користувача, оновлює конфігурацію та відображає відповідні повідомлення.

Клас UserInput має наступні методи:

– метод setRule(rule: String) встановлює значення правила. Він отримує рядок, що представляє правило, як параметр.

– метод setSourceIP(sourceIP: String) встановлює значення IP-адреси джерела. Він отримує рядок, що представляє IP-адресу джерела, як параметр.

– setDestinationIP(destinationIP: String): void: Цей метод встановлює значення IP-адреси призначення. Він отримує рядок, що представляє IP-адресу призначення, як параметр.

– setProtocol(protocol: String): void: Цей метод встановлює значення протоколу. Він отримує рядок, що представляє протокол, як параметр.

– getRule(): String: Цей метод повертає значення правила.

– getSourceIP(): String: Цей метод повертає значення IP-адреси джерела.

– getDestinationIP(): String: Цей метод повертає значення IP-адреси призначення.

– getProtocol(): String: Цей метод повертає значення протоколу.

Діаграма класів для SuricataFilteringAlgorithm на рисунку 2.4

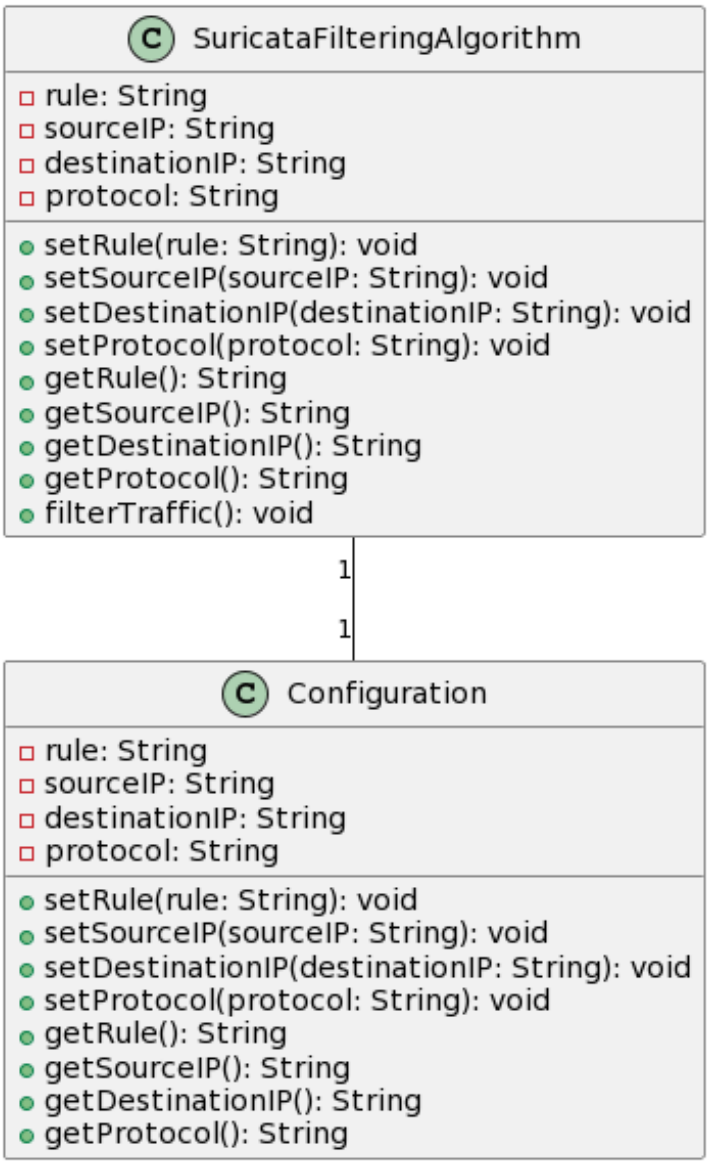

Рисунок 2.4 – Діаграма класів

2.3.2. Сценарій використання (Use Case) для функції запиту:

– Користувач відкриває інтерфейс користувача.

– Користувач вводить параметри правила фільтрації, такі як умови, дія, пріоритет, напрямок, джерело та призначення.

– Інтерфейс користувача передає введені параметри алгоритму обробки правил фільтрації.

– Алгоритм обробки правил фільтрації створює новий об'єкт правила фільтрації з введеними параметрами.

– Алгоритм застосовує створене правило фільтрації до системи Suricata для фільтрації мережевого трафіку.

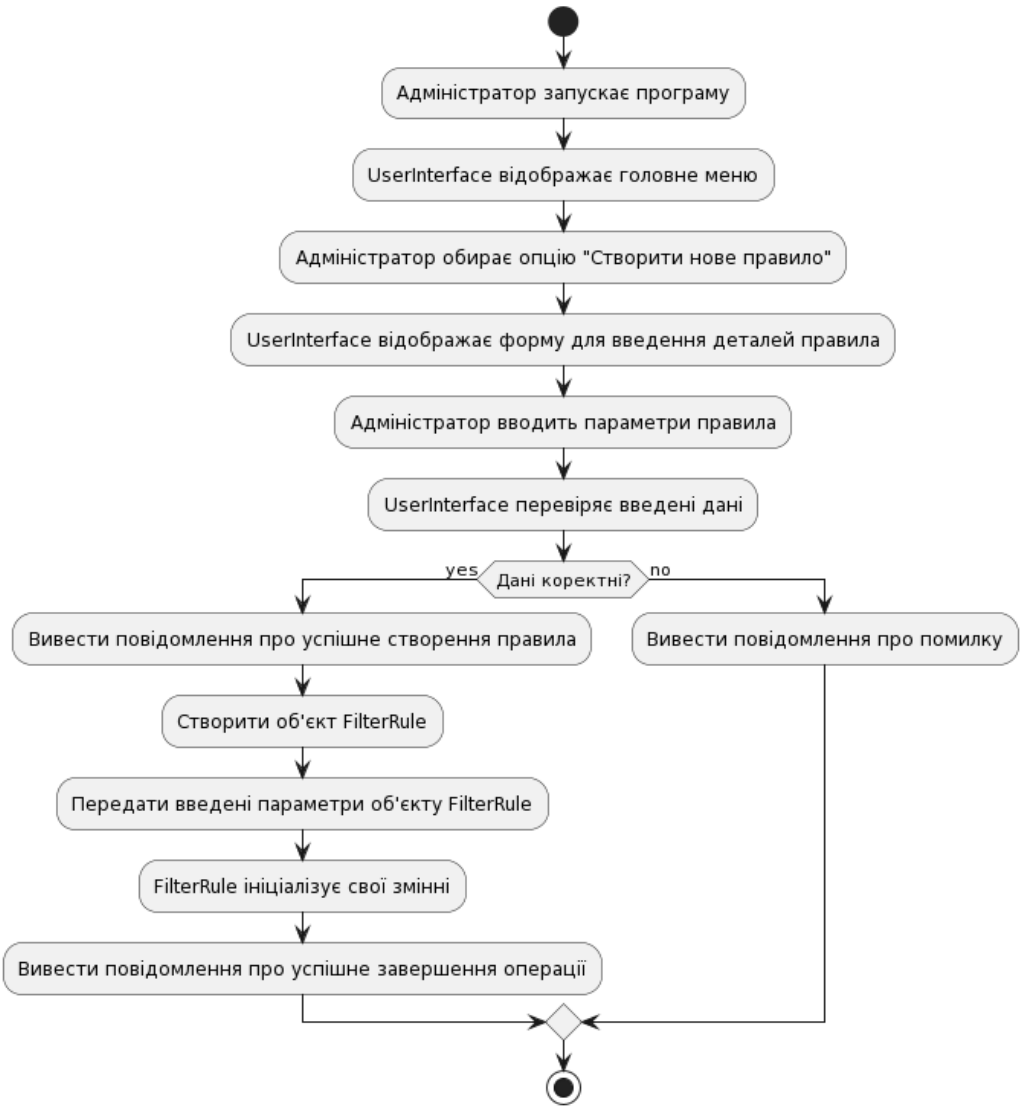

Рисунок 2.5 – алгоритми керування правилами

2.3.3 Опис кроків тестового сценарію "Створення нового правила для опису мережевої атаки":

1. Адміністратор запускає програму інтерфейсу користувача (UserInterface).
2. UserInterface відображає головне меню з доступними опціями.
3. Адміністратор обирає опцію "Створити нове правило".
4. UserInterface відображає форму для введення деталей правила.
5. Адміністратор вводить назву правила, тип атаки, умови фільтрації та інші необхідні параметри.

6. UserInterface перевіряє правильність введених даних і виводить повідомлення про успішне створення правила або про помилку, якщо дані некоректні.
7. Якщо дані коректні, UserInterface передає введені параметри об'єкту FilterRule для створення нового правила.
8. FilterRule отримує передані параметри та ініціалізує свої внутрішні змінні згідно введених даних.
9. Після успішного створення правила, UserInterface виводить повідомлення про успішне завершення операції.
10. Сценарій завершується, програма очікує дії адміністратора.

Розроблено алгоритми керування правилами детекції атак.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Інструментальні засоби побудови системи детекції атак

Проектований засіб IDS повинен забезпечити виявлення, моніторинг та аналіз вторгнень у мережу, а також надати можливість візуалізації та керування правилами. Деталізуємо окремі функції та визначимо компоненти системи:

1. Система Suricata виявлення і запобігання вторгнень (IDS/IPS) аналізує мережевий трафік та ідентифікує потенційні загрози або аномальну активність, що може вказувати на вторгнення. Suricata також може вживати заходів для запобігання атак.

2. Розподілена система зберігання та пошуку даних Elasticsearch забезпечує потужний механізм індексації, зберігання та пошуку даних, що надходять з Suricata. Це дозволяє зберігати, організовувати та швидко відшуковувати дані про виявлені загрози та події.

3. Система збору та обробки логів Logstash використовується для збору, структурування та трансформації даних з різних джерел, включаючи дані від Suricata. Це допомагає нормалізувати дані та підготувати їх для подальшого зберігання та аналізу.

4. Інтерфейс візуалізації та аналізу даних Kibana надає можливості візуалізації та аналізу даних, збережених в Elasticsearch. Вона дозволяє створювати різноманітні графіки, діаграми, панелі та інші візуальні елементи для відображення та аналізу виявлених загроз та подій.

5. Веб-інтерфейс для управління правилами Scirius дозволяє адміністраторам змінювати та налаштовувати правила виявлення в Suricata, щоб адаптувати його до конкретних потреб та вимог безпеки.

6. EveBox: Інтерфейс для відстеження подій та аналізу. EveBox надає можливості перегляду та аналізу подій, виявлених Suricata. Він допомагає адміністраторам більш детально розуміти та аналізувати виявлені загрози та події, а також приймати відповідні заходи для захисту мережі.

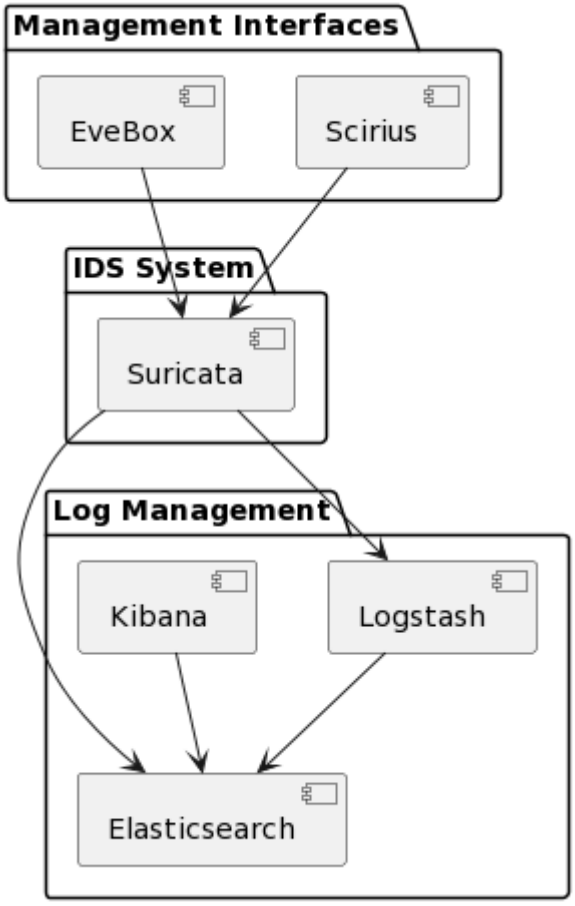

Рисунок 3.1 – UML діаграма структури системи IDS

Elasticsearch - це розподілена система зберігання та аналізу даних, яка базується на пошуковому двигуні Apache Lucene [22]. Вона широко використовується для пошуку, аналізу та візуалізації великих обсягів даних в реальному часі. Забезпечує високу продуктивність та масштабованість.

Відкрите програмне забезпечення Elasticsearch працює на серверах під управлінням операційної системи Linux, а також підтримується на інших платформах. Elasticsearch включає текстовий пошук, геолокаційний пошук, пошук по атрибутам, фасетний пошук, ранжування результатів і багато іншого. Він також підтримує розподілену архітектуру, що дозволяє горизонтальне масштабування та високу доступність даних.

Окрім того, Elasticsearch має багатий екосистемний стек, який включає Kibana [21] для візуалізації та аналізу даних, Logstash для збору, обробки та передачі даних, Beats для моніторингу різних типів даних, і багато інших інструментів, які доповнюють його функціональність.

Реалізуємо алгоритми спроектовані в розділі 2.

Реалізуємо функцію створення нового правила для опису мережевої атаки та передачі його програмі Suricata через API яка описана сценарієм "Створення нового правила для опису мережевої атаки" в п.2.3.3:

```
import requests

def create_filter_rule(rule):
    # Виконати запит API до Suricata для створення правила
    url = "http://suricata-api.example.com/rules"
    headers = {"Content-Type": "application/json"}
    data = {"rule": rule}

    response = requests.post(url, headers=headers, json=data)

    if response.status_code == 200:
        print("Правило було успішно створено.")
    else:
        print("Виникла помилка при створенні правила.")

# Введення параметрів правила
rule = "alert tcp any any -> any 80 (msg:\"HTTP traffic detected\"; content:\"GET \";)"

# Виклик функції для створення правила
create_filter_rule(rule)
```

Використовуємо бібліотеку requests для здійснення HTTP-запитів до API Suricata. Параметр rule містить введене правило, яке буде передано у вигляді JSON-об'єкту. Після виконання запиту, код перевіряє статус відповіді і виводить відповідне повідомлення про успішне створення правила або про помилку.

Структура правила для опису мережевої атаки, яке може бути збережене у вигляді JSON-об'єкту, може мати наступні поля:

1. name (рядок): Назва правила або його ідентифікатор;
2. description (рядок): Опис мережевої атаки, яку правило виявляє;
3. protocol (рядок): Протокол, який використовується у мережевій атакі (наприклад, "TCP", "UDP", "ICMP" і т.д.);
4. sourceIP (рядок): IP-адреса або діапазон IP-адрес, які представляють джерело атаки;

5. destinationIP (рядок): IP-адреса або діапазон IP-адрес, які представляють ціль атаки;

6. sourcePort (число): Порт або діапазон портів джерела атаки.

7. destinationPort (число): Порт або діапазон портів цілі атаки.

8. content (рядок): Вміст, який повинен бути присутнім у мережевому пакеті для виявлення атаки.

9. action (рядок): Дія, яка виконується, коли виявлено атаку (наприклад, "alert", "drop", "reject" і т.д.).

Нижче наведений приклад JSON-структури для правила опису мережевої атаки:

```
json
{
  "name": "HTTP traffic detected",
  "description": "Detects HTTP traffic on port 80",
  "protocol": "TCP",
  "sourceIP": "any",
  "destinationIP": "any",
  "sourcePort": "any",
  "destinationPort": 80,
  "content": "GET ",
  "action": "alert"
}
```

Цей JSON-об'єкт включає всі основні поля, необхідні для опису мережевої атаки. Ви можете додати додаткові поля або використовувати розширену структуру в залежності від ваших потреб і специфікацій IDS системи.

Код функції яка буде запитувати в користувача з командної стрічки параметри правила і передавати в функцію create_filter_rule(rule)

```
def get_filter_rule_from_user():
    rule = {}
```

Запитати параметри правила від користувача

```
    rule['name'] = input("Введіть назву правила: ")
    rule['description'] = input("Введіть опис правила: ")
    rule['protocol'] = input("Введіть протокол: ")
```

```
    rule['sourceIP'] = input("Введіть джерело IP: ")
    rule['destinationIP'] = input("Введіть ціль IP: ")
    rule['sourcePort'] = input("Введіть джерело порту: ")
    rule['destinationPort'] = input("Введіть ціль порту: ")
    rule['content'] = input("Введіть вміст: ")
    rule['action'] = input("Введіть дію: ")

    return create_filter_rule(rule)
```

3.2 Розгортання системи детекції атак

Встановлення системи Suricata на Ubuntu Linux. Спочатку оновимо пакетні списки:

```
sudo apt update
```

2. Встановимо Suricata, виконавши команду:

```
bash
sudo apt install suricata
```

3. Після встановлення Suricata потрібно налаштувати файл конфігурації. Файл `/etc/suricata/suricata.yaml` у текстовому редакторі:

```
bash
sudo nano /etc/suricata/suricata.yaml
```

4. У файлі `suricata.yaml` налаштуємо параметри. Особливу увагу слід звернути на наступні параметри:
   - `HOME_NET`: Вкажіть мережу, яка є внутрішньою (захищеною) мережею.
   - `EXTERNAL_NET`: Вкажіть мережу, яка є зовнішньою (небезпечною) мережею.
   - `INTERFACE`: Вкажіть мережевий інтерфейс, на якому буде прослуховуватись Suricata.

   Ви можете змінювати інші налаштування за своїми потребами.
5. Збережемо зміни в файлі `suricata.yaml` та закрийте редактор.
6. Запустимо Suricata:

```
bash
sudo suricata -c /etc/suricata/suricata.yaml -i
<інтерфейс>
```

де <інтерфейс> - це мережевий інтерфейс, на якому буде прослуховуватись Suricata.

7. Suricata почне прослуховувати вказаний мережевий інтерфейс і аналізувати мережевий трафік, виявляючи потенційні загрози.

Після встановлення ви можна додати правила виявлення загроз та настроїти журналування, вивід та інші параметри. На рисунку показана вею інтерфейс для керування Suricata.

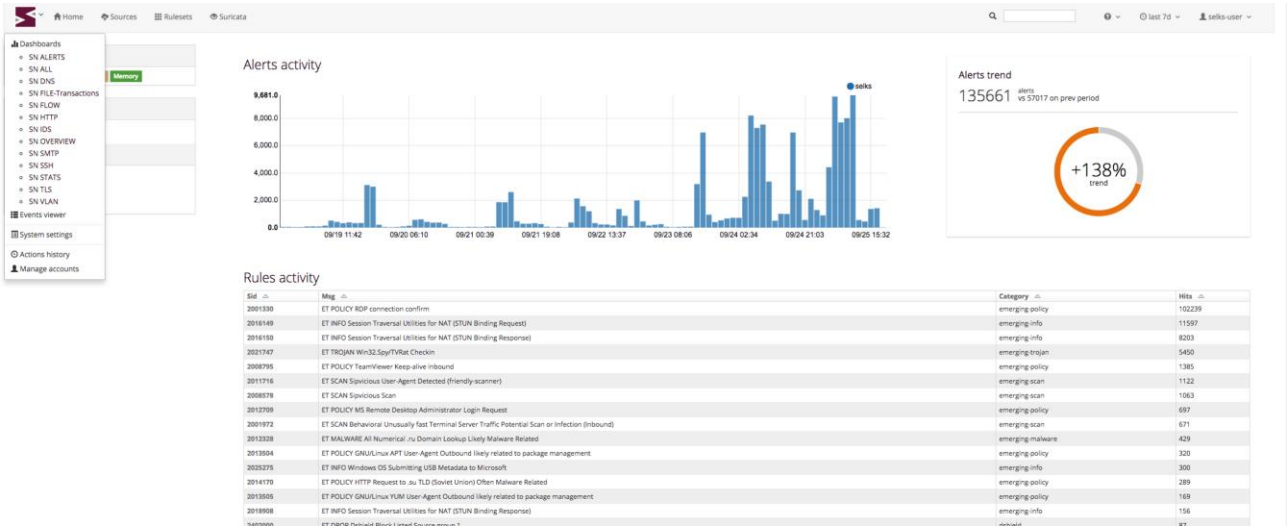

Рисунок 3.1 - Веб-інтерфейс Scirius

3.3 Тестування системи детекції атак

Методика тестування IDS систем зазвичай включає наступні етапи.

1. Планування тестування. Визначаються цілі тестування, обсяг ресурсів, доступних для тестування, тестові сценарії та вимоги до тестового середовища.

2. Створення тестового середовища. Розгортання тестового середовища, яке включає в себе встановлення IDS системи, мережевих пристроїв, серверів і клієнтських машин.

3. Генерація тестового трафіку. Для тестування IDS системи можуть використовуватися різні інструменти для генерації тестового трафіку, такі як

інструменти для створення мережевих пакетів або спеціальні програми для генерації атак.

4. Запуск тестових сценаріїв. Тестові сценарії включають різні типи атак, спроби злому, вторгнення і т. д. За допомогою цих сценаріїв перевіряються реакція IDS системи на потенційні загрози.

5. Збір і аналіз результатів. Після завершення тестування збираються дані про виявлені атаки, хибні спрацювання, ефективність IDS системи тощо. Ці дані аналізуються для визначення ефективності і надійності системи.

Щодо в мережах з маршрутизатором MikroTik тестування включає додаткові етапи:

1. Створення мережевої інфраструктури, включаючи маршрутизатор MikroTik, комутатори, сервери і клієнтські комп'ютери.

2. Встановлення та конфігурування IDS системи на одній або декількох машинах у мережі.

Структурна схема тестової комп'ютерної мережі включає такі компоненти:

- Клієнтські комп'ютери: Комп'ютери, з яких генерується тестовий трафік або виконуються атаки на мережу.
- Сервери: Сервери, які використовуються для розгортання IDS системи, збору логів, аналізу результатів і т. д.
- Мережеві пристрої: Маршрутизатори, комутатори та інші мережеві пристрої, які забезпечують з'єднання і передачу даних в мережі.
- IDS система: Система виявлення і запобігання вторгненням, розташована на окремих серверах або вбудована безпосередньо в мережеві пристрої.
- Тестові інструменти: Інструменти для генерації тестового трафіку або виконання атак на мережу, які використовуються для тестування IDS системи.

У цій схемі комп'ютерної мережі два клієнтських комп'ютери, які підключені до мережевого комутатора. Маршрутизатор забезпечує з'єднання з мережею, а IDS система розташована після маршрутизатора для виявлення та моніторингу потенційних загроз у мережі. Також в мережі є тестовий

інструмент та сервер для збору результатів тестування. Вузол "Генератор трафіку", який відповідає за генерацію тестових пакетів і передачу їх у мережу. Цей вузол може використовуватися для симуляції різних типів атак або навантаження на систему, щоб перевірити ефективність роботи IDS системи.

Деталізований опис функцій та обладнання для кожного блоку:

1. Клієнтський комп'ютер:

– Програмне забезпечення: операційна система (наприклад, Windows, macOS, Linux) та додаткове програмне забезпечення, яке використовується на клієнтському комп'ютері (браузери, додатки тощо).

– Апаратне забезпечення: клієнтський комп'ютер з необхідною конфігурацією (процесор, пам'ять, мережеві інтерфейси тощо).

2. Мережевий комутатор:

– Програмне забезпечення: програмне забезпечення комутатора для керування мережевим трафіком та передачі даних між вузлами.

– Апаратне забезпечення: мережевий комутатор з необхідною кількістю портів та підтримкою необхідних мережевих протоколів.

3. Маршрутизатор:

– Програмне забезпечення: Операційна система маршрутизатора (наприклад, RouterOS в MikroTik) та програмне забезпечення для керування маршрутизацією та мережевими протоколами.

– Апаратне забезпечення: Маршрутизатор з необхідною конфігурацією (процесор, пам'ять, мережеві інтерфейси тощо).

4. IDS система:

– Програмне забезпечення: IDS система (Suricata) з вбудованими алгоритмами виявлення інтра- та екстранет-атак, обробкою трафіку та генерацією сповіщень про події.

– Апаратне забезпечення: Сервер або спеціалізована апаратна платформа з високою продуктивністю та достатніми ресурсами для обробки мережевого трафіку.

5. Тестовий інструмент:

– Програмне забезпечення для тестування IDS системи, включаючи засоби генерації тестового трафіку, атак та аналізу реакції IDS системи.

– Апаратне забезпечення: Комп'ютер або сервер з необхідною конфігурацією для виконання тестових сценаріїв та взаємодії з іншими вузлами мережі.

6. Генератор трафіку:

– Програмне забезпечення для генерації тестового мережевого трафіку з різними параметрами (типи пакетів, інтенсивність, розмір тощо).

– Апаратне забезпечення: Комп'ютер або сервер з достатньою продуктивністю для генерації великого обсягу трафіку.

7. Сервер:

– Програмне забезпечення: Серверне програмне забезпечення, яке забезпечує функціональність, доступність і роботу мережевих служб (наприклад, веб-сервер, база даних тощо).

– Апаратне забезпечення: Сервер з високою продуктивністю та достатніми ресурсами для обробки запитів та надання послуг.

Зобразимо узагальнену схему тестового середовища для розроблених засобів на рисунку 3.3.

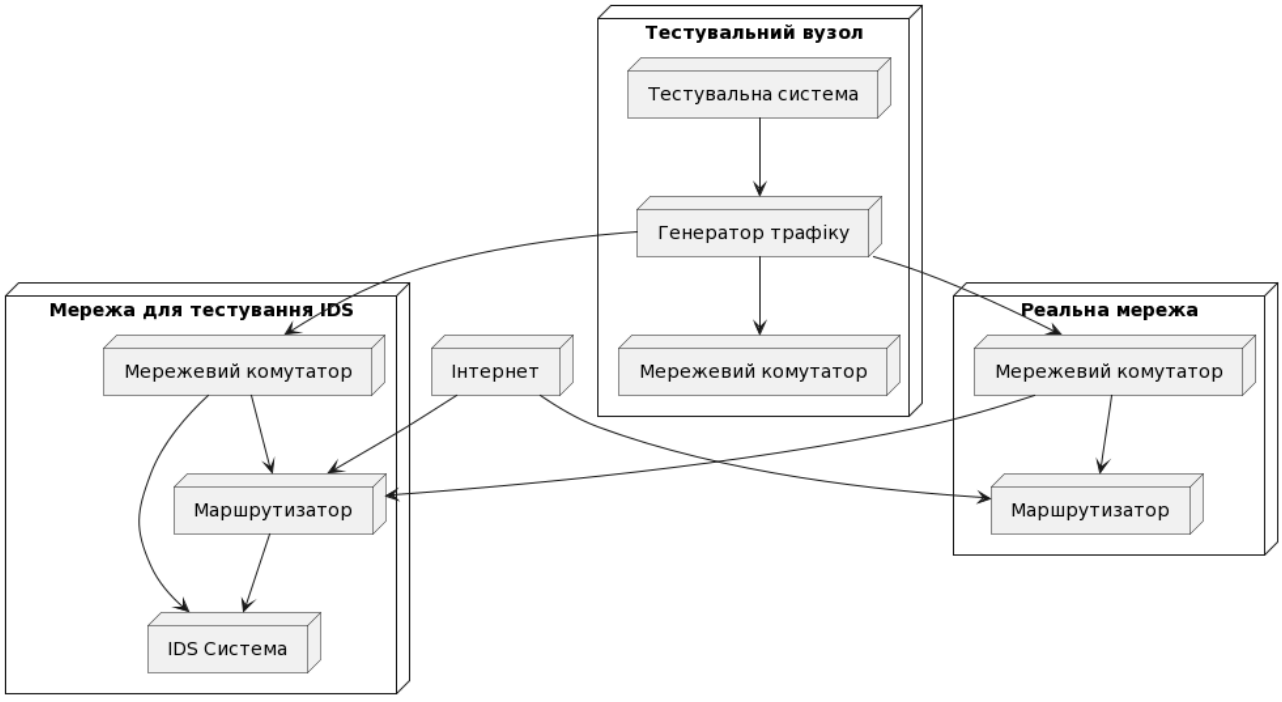

Рисунок 3.3 - Узагальнена схему тестового середовища

Діаграма розгортання має практичну цінність для тестування і ілюстрації IDS систем, оскільки вона надає візуальне представлення архітектури мережі та взаємозв'язків між її компонентами. Основні переваги використання діаграми розгортання описано нижче.

1. Візуалізація архітектури:. Діаграма надає зрозумілу візуальну ілюстрацію розташування компонентів системи IDS у мережі тестування та реальній мережі. Це допомагає зрозуміти, як різні компоненти системи взаємодіють між собою та з іншими елементами мережі.

2. Діаграма розгортання допомагає виявити залежності між компонентами системи та інфраструктурою мережі. Це дозволяє враховувати взаємовплив різних елементів на ефективність IDS системи.

3. Діаграма може допомогти в оцінці надійності системи IDS шляхом визначення резервування та розподілу ресурсів мережі. Вона дозволяє ідентифікувати потенційні проблеми, такі як односпрямованість трафіку або одиночна точка відмови.

4. Визначення тестових сценаріїв. Deployment діаграма може служити основою для визначення тестових сценаріїв та наборів тестів для перевірки працездатності і ефективності IDS системи. Вона надає контекст для розробки тестових кейсів, які можуть включати симуляцію різних видів трафіку, атак та інших сценаріїв.

5. Deployment діаграма може служити засобом комунікації між розробниками IDS системи, тестувальними інженерами, адміністраторами мережі та іншими зацікавленими сторонами. Вона допомагає уникнути непорозумінь та сприяє збільшенню згоди щодо архітектури і конфігурації системи.

Деталізуємо програмне забезпечення програмного комплексу. Діаграма демонструє мережеву і апаратну архітектуру для тестування IDS системи в мережах з маршрутизатором MikroTik. Клієнтський пристрій взаємодіє з програмним забезпеченням UserInterface, яке отримує введені користувачем дані через UserInput. UserInterface передає ці дані в SuricataFilteringAlgorithm,

який реалізує алгоритм фільтрації Suricata. Тестувальний інструмент TestingTool використовує TrafficGenerator для генерації тестового трафіку і IDSSystem для моніторингу та аналізу цього трафіку. Сервер надає послуги, а база даних використовується для зберігання даних. Мережеві компоненти, такі як маршрутизатор MikroTik, маршрутизатор, комутатор, пристрій безпеки і мережевий фаєрвол, забезпечують мережеву інфраструктуру для передачі трафіку.

Детальна схема розгортання системи детекції вторгнень зображена на кресленні КР.КІ.9499967.00.00.001.С1.

Для генерації трафіку та тестування системи IDS існує кілька популярних програмних засобів:

1. Засіб Scapy [24] дозволяє створювати, відправляти та перехоплювати мережеві пакети з власними налаштуваннями.

2. D-ITG (Distributed Internet Traffic Generator) [25] є інструментом для генерації реалістичного мережевого трафіку з різними протоколами та параметрами. Він може бути використаний для тестування пропускної здатності мережі та перевірки реакції IDS системи на різні типи трафіку.

3. ПЗ hping є універсальним інструментом для відправки налаштованих мережевих пакетів та аналізу відповіді. Він дозволяє генерувати кастомний трафік з різними протоколами та параметрами для тестування IDS системи.

4. ПЗ tcpreplay дозволяє відтворювати збережені мережеві пакети з файлів у реальному часі. Це дозволяє використовувати реальний трафік для тестування IDS системи, роблячи його більш реалістичним та варіативним.

5. ПЗ Nemesis є інструментом для створення та відправки налаштованих мережевих пакетів з різними протоколами. Він дозволяє генерувати тестовий трафік для виявлення вразливостей та перевірки реакції IDS системи.

Для тестування розробленого ПЗ напишемо тест для функції get_filter_rule_from_user(). Оголошуємо функцію і створюємо вхідні дані

```
def test_get_filter_rule_from_user():
    # Підмінити ввід користувача замість
    input_data = [
```

```
        "Правило 1",  # Назва правила
        "Опис правила 1",  # Опис правила
        "TCP",  # Протокол
        "192.168.0.1",  # Джерело IP
        "192.168.0.2",  # Ціль IP
        "8080",  # Джерело порту
        "80",  # Ціль порту
        "Attack Payload",  # Вміст
        "Block",  # Дія
    ]
    # Зберігти оригінальну функцію input
    original_input = __builtins__.input
    # Перезаписати функцію input для повернення даних з input_data
    __builtins__.input = lambda _: input_data.pop(0)
```

Виклик функції та збереження результату

```
    result = get_filter_rule_from_user()
    # Перевірка результату
    expected_result = {
        'name': 'Правило 1',
        'description': 'Опис правила 1',
        'protocol': 'TCP',
        'sourceIP': '192.168.0.1',
        'destinationIP': '192.168.0.2',
        'sourcePort': '8080',
        'destinationPort': '80',
        'content': 'Attack Payload',
        'action': 'Block'
    }
    assert result == expected_result, "результат не співпадає."
```

Відновити оригінальну функцію input

```
    # Відновити оригінальну функцію input
    __builtins__.input = original_input

    print("Тест пройдено успішно.")
```

Виклик функції тесту

```
test_get_filter_rule_from_user()
```

Код розроблених алгоритмів наводиться у додатку А. В даному розділі обрано мову і середовище програмування. Описано всі етапи розроблення програмного засобу відповідно до технічного завдання.

4 ТЕХНІКО-ЕКОНОМІЧНИЙ РОЗДІЛ

У даному розділі кваліфікаційної роботи проводиться економічне обґрунтування доцільності розробки програмного забезпечення системи детекції вторгнень на основі обладнання Mikrotik. Зокрема, здійснюється розрахунок витрат на розробку даного програмного продукту, експлуатаційних витрат, ціни на споживання проектного рішення, визначаються показники економічної ефективності нового програмного продукту, обґрунтовуються відповідні висновки.

Розроблений електронний посібник призначений для використання студентами другого курсу спеціальності «Комп'ютерна інженерія».

4.1 Розрахунок витрат на розробку програмного забезпечення

Витрати на розробку і впровадження програмних засобів ($К$) включають:

$$К = К_1 + К_2, \tag{4.1}$$

де $К_1$ – витрати на розробку програмних засобів, грн.;

$К_2$ – витрати на відлагодження і дослідну експлуатацію програми рішення задачі на комп'ютері, грн.

Витрати на розробку програмних засобів включають:

- витрати на оплату праці розробників ($В_{ОП}$);
- витрати на відрахування у спеціальні державні фонди ($Вф$);
- витрати на покупні вироби ($Пв$);
- витрати на придбання спецобладнання для проведення експериментальних робіт ($Об$);
- накладні витрати ($Н$);

– інші витрати (*Ів*).

4.1.1 Розрахунок витрат на оплату праці

Витрати на оплату праці включають заробітну плату (ЗП) всіх категорій працівників, безпосередньо зайнятих на всіх етапах проектування. Розмір ЗП обчислюється на основі трудомісткості відповідних робіт у людино-днях та середньої ЗП відповідних категорій працівників.

У розробці проектного рішення задіяні наступні спеціалісти-розробники, а саме: керівник проекту; студент-дипломник (таблиця 4.1).

Таблиця 4.1 – Вихідні дані для розрахунку витрат на оплату праці

Посада виконавців	Погодинна заробітна плата, грн.
Керівник КР	124 грн/год
Студент	17 грн/год

Витрати на оплату праці розробників проекту визначаються за формулою:

$$B_{ОП} = \sum_{i=1}^{N} \sum_{j=1}^{M} n_{ij} \cdot t_{ij} \cdot C_{ij}\,, \qquad (4.2)$$

де n_{ij} – чисельність розробників *i*-ої спеціальності *j*-го тарифного розряду, осіб;

t_{ij} – затрачений час на розробку проекту співробітником *i*-ої спеціальності *j*-го тарифного розряду, год;

C_{ij} – годинна ставка працівника *i*-ої спеціальності *j*-го тарифного розряду, грн.

Середньогодинну ставку працівника розраховуємо за формулою:

$$C_{ij} = \frac{C_{ij}^{0}(1+h)}{РЧ_{i}}, \qquad (4.3)$$

де C_{ij}^{0} – основна місячна заробітна плата розробника *i*-ої спеціальності *j*-го тарифного розряду, грн.;

h – коефіцієнт, що визначає розмір додаткової заробітної плати (при умові наявності доплат);

$РЧ_i$ – місячний фонд робочого часу працівника *i*-ої спеціальності *j*-го тарифного розряду, год. (приймаємо 168 год.).

Результати розрахунку записуємо у таблицю 4.2.

Таблиця 4.2 – Розрахунок витрат на оплату праці

Посада виконавців	Час розробки, год.	Погодинна заробітна плата, грн/год.	Витрати на розробку, грн
Керівник КР	16	124	1984
Студент	234	17	3978
Разом витрати на розробку			5962

4.1.2 Відрахування на соціальні заходи

Відрахування на соціальні заходи для керівника КР включають:

1) ЄСВ (єдиний соціальний внесок). Він становить 22% від заробітної плати. Оскільки заробіток керівника КР становить 1984 грн, то ЄСВ буде становити:

$$1984 \cdot 0{,}22 = 436{,}48 \text{ грн};$$

2)ПДФО (податок на доходи фізичних осіб). Він становить 18%. ПДФО буде становити:

$$1984 \cdot 0{,}18 = 357{,}12 \text{ грн};$$

3)ВЗ (військовий збір). Він становить 1,5%. ВЗ буде становити:

$$1984 \cdot 0{,}015 = 29{,}76 \text{ грн.}$$

Працівнику чистими має бути перераховано за місяць:

$$1984 - 436{,}48 - 357{,}12 - 29{,}76 = 1160{,}64 \text{ грн.}$$

Оскільки до розрахунку загального місячного (річного) оподаткованого доходу платника податку не включається, зокрема, сума стипендій України, призначених законом, постановами Верховної Ради України, указами Президента України, то заробіток студента залишається незмінним – 3978 грн.

4.1.3 Розрахунок витрат на матеріали та комплектуючі

У таблиці 4.3 наведений перелік купованих виробів і розраховані витрати на них.

Таблиця 4.3 – Розрахунок витрат на матеріали та комплектуючі

Найменування	Виробник (модель)	Одиниці вимірювання	Кількість	Ціна за одиницю, грн	Сума, грн
Маршрутизатор	MikroTik RouterBOARD RB951Ui-2HnD	шт.	1	2482	2482
Разом					2482

4.1.4 Витрати на використання комп’ютерної техніки

Якщо для розробки КС використовується електрообладнання, то необхідно розрахувати витрати на електроенергію за формою, наведеною в таблиці 4.4.

Таблиця 4.4 – Розрахунок витрат на використання комп'ютерної техніки

Назва устаткування	Паспортна потужність	Коефіцієнт використання потужності	Час роботи обладнання, год	Ціна електроенергії, кВт год грн	Сума, грн
Ноутбук	0,17	0,8	234	1,44	57,2
Разом витрати на електроенергію					57,2

Накладні витрати проектних організацій включають три групи видатків: витрати на управління, загальногосподарські витрати, невиробничі витрати. Вони розраховуються за встановленими відсотками до витрат на оплату праці. Середньостатистичний відсоток накладних витрат приймемо 150% від заробітної плати:

$$H = 1{,}5 \cdot 5962 = 8943 \text{ грн.}$$

Інші витрати є витратами, які не враховані в попередніх статтях. Вони становлять 10% від заробітної плати:

$$I = 0{,}1 \cdot 5962 = 596{,}2 \text{ грн.}$$

Витрати на розробку програмного забезпечення складають:

$$K_1 = B_{ОП} + B_Ф + B_{ПВ} + H + I, \quad (4.4)$$

$$K_1 = 1847{,}6 + 378{,}76 + 187 + 2771{,}4 + 184{,}76 = 5369{,}86 \text{ грн.}$$

Витрати на відлагодження і дослідну експлуатацію програмного продукту визначаємо за формулою:

$$K_2 = S_{м.г.} \cdot t_{від}$$

де $S_{м.г.}$ – вартість однієї машино-години роботи ПК, грн./год.

$t_{від}$ – комп'ютерний час, витрачений на відлагодження і дослідну експлуатацію створеного програмного продукту, год.

Загальна кількість днів роботи на комп'ютері дорівнює 30 днів. Середній щоденний час роботи на комп'ютері – 2 години. Вартість години роботи комп'ютера дорівнює 5,2 грн. Тому

$$K_2 = 5{,}2 \cdot 80 = 416 \text{ грн.}$$

На основі отриманих даних складаємо кошторис витрат на розробку програмного забезпечення (таблиця 4.5).

Таблиця 4.5 – Кошторис витрат на розробку програмного забезпечення

№ п/п	Найменування витрат	Сума витрат, грн.
1	Витрати на оплату праці	1847,6
2	Відрахування у спеціальні державні фонди	378,76
3	Витрати на куповані вироби	187
4	Накладні витрати	2771,4
5	Інші витрати	184,76
6	Витрати на відлагодження і дослідну експлуатацію програмного продукту	416
Разом		5785,52

4.2 Визначення витрат на експлуатацію програмного продукту

Для оцінки економічної ефективності розроблюваного програмного продукту слід порівняти його з аналогом, тобто існуючим програмним забезпеченням ідентичного функціонального призначення.

Розрахуємо річні поточні витрати на експлуатацію програмного забезпечення $B_{еПК}$, які визначаються за формулою:

$$B_{еПК} = B_a + B_e + B_{рем} + B_{дк} + B_i, \quad (4.5)$$

де B_a – річні відрахування на амортизацію,

B_e – річні витрати на електроенергію для ПК,

$B_{рем}$ – річні витрати на ремонт ПК,

$B_{дк}$ – річні витрати на додаткові комплектуючі ПК,

B_i – інші витрати.

Обчислимо кожен з цих показників.

Суму річних амортизаційних відрахувань визначаємо за такою формулою:

$$B_a = Ц_{ПК} \cdot H_a, \tag{4.6}$$

де $Ц_{ПК}$ – балансова вартість ПК,

H_a – норма амортизаційних відрахувань (дорівнює 15% у квартал).

Балансову вартість ПК розраховуємо за формулою:

$$Ц_{ПК} = Ц_р \cdot (1 + K_{ун}), \tag{4.7}$$

де $Ц_р$ – ринкова вартість ПК,

$K_{ун}$ – коефіцієнт, що враховує витрати на установку й налагодження ПК (приймається рівним 12%).

Ринкова вартість ПК (Ноутбук ASUS ZenBook Duo 14 UX482EG-HY419W (90NB0S51-M003H0) Celestial Blue) становить 46000 грн. Отже, якщо $Ц_р$ = 46000, $K_{ун}$ = 0,12, то за формулою (4.7) маємо:

$$Ц_{ПК} = 46000 \cdot (1 + 0{,}12) = 51520 \text{ грн.}$$

Обчислимо норму амортизаційних відрахувань H_a. Оскільки відомо, що амортизаційні відрахування дорівнюють 15% у квартал, то

$$H_a = 4 \cdot 15\% = 60\% = 0{,}6.$$

Отже, за формулою (4.6) маємо річні відрахування на амортизацію:

$$B_a = 51520 \cdot 0{,}6 = 30912 \text{ грн.}$$

Витрати на електроенергію, що споживає ПК, визначаємо за формулою:

$$B_e = P_{ПК} \cdot Ф_{ПК} \cdot Ц_e \cdot P_{ів}, \qquad (4.8)$$

де $P_{ПК}$ – паспортна потужність ПК,

$Ф_{ПК}$ – річний фонд корисного часу роботи ПК,

$Ц_e$ – вартість 1 кВт/год електроенергії,

$P_{ів}$ – коефіцієнт інтенсивного використання ПК (0,7 – 1).

Врахувавши, що $P_{ПК}$ = 0,8, $Ф_{ПК}$ = 1843 год, $Ц_e$ = 1,44 кВт·год, $P_{ів}$ = 0,9, отримуємо:

$$B_e = 0{,}8 \cdot 1843 \cdot 1{,}44 \cdot 0{,}9 = 1911 \text{ грн.}$$

Витрати на поточний і профілактичний ремонт $B_{рем}$ приймаються рівними 6% від вартості ПК:

$$B_{рем} = Ц_{ПК} \cdot 0{,}06.$$

Отже,

$$B_{рем} = 51520 \cdot 0{,}06 = 3091{,}2 \text{ грн.}$$

Витрати на додаткові комплектуючі $B_{дк}$ – витрати необхідні для забезпечення експлуатації ПК, приймаються рівними 2% від вартості ПК:

$$B_{дк} = Ц_{ПК} \cdot 0{,}02.$$

Отже,

$$B_{дк} = 51520 \cdot 0{,}02 = 1030{,}4 \text{ грн.}$$

Інші витрати, тобто непрямі витрати пов'язані з експлуатацією ПК (приймаються рівними 5-10% від вартості ПК). Приймемо їх 5%:

$$B_i = Ц_{ПК} \cdot 0{,}05.$$

Отже,

$$B_i = 51520 \cdot 0{,}05 = 2576 \text{ грн.}$$

Отже, підставивши усі обчислені витрати у формулу (4.5), отримаємо річні поточні витрати на експлуатацію програмного забезпечення:

$$B_{еПК} = 1911 + 30912 + 3091{,}2 + 1030{,}4 + 2576 = 39520{,}6 \text{ грн.}$$

4.3 Розрахунок ціни програмного продукту

Ціна споживання програмного продукту – це витрати на придбання і експлуатацію програмного засобу за весь період його служби:

$$Ц_{С(П)} = Ц_П + B_{(E)NPV}, \quad (4.9)$$

де $Ц_П$ – ціна придбання програмного продукту, грн.

$$Ц_П = K(1 + \frac{П_Р}{100}) + K_0 + K_к,$$

де K – кошторисна вартість;

$П_р$ – рентабельність;

K_0 – витрати на прив'язку та освоєння програмного засобу на конкретному об'єкті, грн.;

$К_к$ – витрати на доукомплектування технічних засобів на об'єкті, грн. Зважаючи на вищеописане, розрахуємо ціну програмного засобу

$$Ц_П = 5785,52 \cdot (1 + 0,3) = 7521,2 \text{ грн.}$$

Вартість витрат на експлуатацію проектного продукту (за весь час його експлуатації), в грн. обчислюється так:

$$B_{enpv} = \sum_{t=0}^{T} \frac{B_{ЕП}}{(1+R)^t}, \quad (4.10)$$

де $B_{ЕП}$ – річні експлуатаційні витрати, грн.;

T – термін служби програмного засобу, років;

R – річна ставка проценту банку.

Розрахуємо витрати на експлуатацію для розробленого програмного продукту та його аналогу:

$$B_{enpv} = \sum_{t=1}^{5} \frac{4299,12}{(1+0,08)^t} = 17200,15 \text{ грн,}$$

$$B_{enpva} = \sum_{t=1}^{5} \frac{6448,68}{(1+0,08)^t} = 25800,2 \text{ грн.}$$

Тоді ціна споживання для розробленого програмного продукту та його аналогу становитиме:

$$Ц_{С(П)} = 7521,2 + 17200,15 = 24721,35 \text{ грн,}$$

$$Ц_{С(П)_a} = 6500 + 25800,2 = 32300,2 \text{ грн.}$$

У наступному підрозділі проведемо аналіз економічної ефективності розробки програмного продукту.

4.4 Визначення показників економічної ефективності

За міжнародними стандартами для оцінки ефективності розробки ПЗ застосовують такі показники:

– внутрішня норма дохідності;
– чистий приведений дохід;
– рентабельність;
– термін окупності.

Показник внутрішньої дохідності характеризує величину чистого прибутку (чистого валового доходу), що припаде на одиницю інвестиційних вкладень у кожному часовому інтервалі життєвого циклу проекту.

Розрахунок цього показника виконується за такою формулою:

$$\sum_{i=0}^{T} \frac{D_i}{(1+q)^i} - \sum_{i=0}^{T} \frac{K_i}{(1+q)^i} = 0 \qquad (4.11)$$

де D_i – дохід (прибуток) у i-му періоді;

K_i – інвестиційні вкладення в i-му періоді з урахуванням інфляційних процесів;

i – періоди виконання і впровадження проекту;

T – загальний період (тривалість) життєвого циклу проекту;

q – показник внутрішньої норми дохідності.

Показник інвестиційних вкладень з урахуванням інфляційних процесів обчислюємо за формулою:

$$K_i = \varphi_i \cdot R_i, \qquad (4.12)$$

де φ_i – коефіцієнт інфляції на поточний період;

R_i – інвестиційні платежі в i-му періоді (капітальні вкладення).

Отже,

$$K_i = 1{,}076 \cdot 70000 = 75320 \text{ грн},$$

де φ_i =107,6% (коефіцієнт інфляції поданий в таблиці на 2022 рік в сфері ІТ)

R_i = 70000 грн.

Дохід від розробки ПЗ у *i*-му періоді розраховуємо за формулою:

$$D_i = J_i\,(B_i - C_i), \qquad (4.13)$$

де B_i – ціна продажу програмного продукту в *i*-му періоді;

C_i – собівартість програмного продукту (фактично дорівнює сумі витрат на розробку ПЗ);

J_i – кількість ПЗ.

Отже,

$$D_i = 1 \cdot (80260 - 64208) = 16052 \text{ грн},$$

де B_i = 80260 грн,

C_i = 64208 грн,

J_i = 1.

Вартість продажу розробленого продукту розраховують за формулою:

$$B_i = B_{заг} \cdot (1 + p/100), \qquad (4.14)$$

де p – середній рівень рентабельності на поточний період.

Отже,

$$B_i = 64208 \cdot (1 + 25/100) = 80260 \text{ грн},$$

де p = 25%.

Показник рентабельності інвестицій. У практиці середнього бізнесу для визначення ефективності проектних рішень широко використовується показник

рентабельності інвестицій. Економічний зміст – характеризує частку чистого приведеного доходу, що припадає на одиницю дисконтованих в період життєвого циклу проекту інвестиційних вкладень:

$$p = \frac{\sum_{i=0}^{T} \frac{D_i}{(1+q)^i}}{\sum_{i=0}^{T} \frac{K_i}{(1+q)^i}} - 1 > 0. \qquad (4.15)$$

У ринкових умовах при ціновій політиці, що змінюється, показник терміну окупності є одним з головних для підприємств. Він визначається на основі величини капітальних витрат по періодах розробки програмного продукту та величини фактичних чи прогнозних доходів:

$$\sum_{i=0}^{T} K_i = \sum_{i=0}^{T} D_i, \qquad (4.16)$$

де T – термін окупності,

D_i – дохід (прибуток) у поточному періоді,

K_i – капітальні витрати у поточному періоді.

Економічна ефективність полягає у відношенні результату від розробленого програмного продукту до затрачених ресурсів:

$$E = D_i / B_{заг}.$$

Отже,

$$E = 16052 / 64208 = 0{,}15.$$

Тоді термін окупності можна розрахувати за такою формулою:

$$T = 1 / E.$$

Отже,

$$T = 1/0,15 = 6,6 \text{ років.}$$

В даному розділі проведено розрахунок витрат на розробку програмного забезпечення. Враховуючи основні економічні показники, що стосуються розробки програмного продукту, можна зробити висновок, щодо доцільності запропонованої розробки. Отримано економічний ефект від розробки програмного продукту 0,15, а термін окупності капітальних вкладень 6,6 років, що є меншим 10 років, то розробка є економічно вигідною та конкурентоздатною на ринку подібних ІТ продуктів.

ВИСНОВКИ

1. Проведено аналіз функцій та структури систем детекції вторгнень. Було досліджено різні методи і технології, включаючи розпізнавання аномалій, сигнатурний аналіз та машинне навчання. Систематизовано структуру правил детекції вторгнень.

2. Проведено аналіз специфіки роботи маршрутизаторів Mikrotik, включаючи їх можливості з обробки мережевого трафіку, доступні функції безпеки. Для реалізації системи використано команди RouterOS та функцію дзеркалювання трафіку на окремий порт маршрутизатора.

3. Розроблено алгоритми керування правилами детекції атак, які враховують специфіку мережі і потенційні загрози. Створено об’єктну модель ПЗ, структуру правила в JSON форматі.

4. Була створена структура мережі з системою IDS, включаючи налаштування маршрутизаторів, комутаторів та інших мережевих пристроїв. В якості компонетів взято системи Suricata, Elasticsearch, веб-інтерфейс Scirius. Систему IDS встановлено на ОС Linux Ubuntu. Розроблено UML діаграму розгортання системи.

5. Алгоритми програмно реалізовані з використанням мови програмування Python та бібліотекb requests, що дозволило запустити систему детекції вторгень на практиці. Після реалізації було проведено тестування алгоритмів керування правилами детекції вторгень, що дозволило оцінити її ефективність та надійність.

6. В результаті проведених досліджень та розробок, була створена система детекції вторгень на основі обладнання Mikrotik, ПЗ Suricata яка дозволяє виявляти потенційні загрози в мережі та реагувати на них, забезпечуючи підвищену безпеку та захист мережевої інфраструктури.

7. Обґрунтовано техніко-економічні показники ефективності розробки програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Козловський О. В., Пліш В.В Засоби забезпечення безпеки мереж на основі правил. VII Науково-практична конференція молодих вчених і студентів «Інтелектуальні комп'ютерні системи та мережі». 23 травня 2023 р. Тернопіль. Україна. с. 17
2. Deploying Applications, Services, and Components. 15.11.2016 - [Електронний ресурс] Режим доступу - https://msdn.microsoft.com/uk-ua/library/wtzawcsz.aspx
3. Barnard Robert. Intrusion Detection Systems 2nd Edition, Butterworth-Heinemann. 1988. 462 p
4. Когут Юрій. Книга Цифрова трансформація економіки та проблеми кібербезпеки. Консалтингова компанія Сідкон, 2021. 368 с.
5. Brotherston Lee. Defensive Security Handbook: Best Practices for Securing Infrastructure 1st Edition. O'reilly Media, 2017. 274 p.
6. Англо-український тлумачний словник з обчислювальної техніки, Інтернету і програмування. — Вид. 1— К.: Видавничий дім «СофтПрес», 2005. — 552 с.
7. Emerging Threats https://rules.emergingthreatspro.com/open/
8. Suricata Rules. Suricata User Guide. https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Suricata_Rules
9. Suricata Rules https://docs.suricata.io/en/suricata-6.0.12/rules/index.html.
10. McCabe J. Network Analysis, Architecture, and Design. Third edition. / James D. McCabe - Morgan Kaufmann, 2007. 495 p.
11. Яковина В.С. Основи безпеки комп'ютерних мереж: Навчальний посібник / За ред. Д.В. Федасюка. Львів: НВФ "Українські технології", 2008. 396 с.
12. The Cisco Learning Network URL: https://learningnetwork.cisco.com
13. Internet Routing Architectures 2nd Edition/ Sam Halabi, Danny McPherson Cisco Press, 2000. 528 p.
14. Документація з настройки обладнання фірми Cisco. [Електронний ресурс]: Режим доступу http://www.cisco.com
15. Saad, A., Khan, S.A., & Mahmood, A. (2018). A multi-objective evolutionary artificial bee colony algorithm for optimizing network topology design. Swarm Evol. Comput., 38, p.187-201.
16. Zaychenko Y.P., Zaychenko H., Hamidov G. Structure optimization of new generation networks. 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), 2017, p. 1-5.

17. Ahmed, S., & Ahmed, J. A Parallel Approach to Solve Minimum Spanning Tree Problem in Network Routing. 2017

18. Witt, R.M., & Emrich, J. Design considerations of a cable wiring system for a new medical center to support a future medical imaging system. 'Medical Imaging VI: PACS Design and Evaluation', SPIE, 1992. p. 486 -491

19. Witt, R.M., Gibbs, T., & Holden, R.W. Intercampus network of the Department of Radiology, School of Medicine, Indiana University. Medical Imaging. SPIE,1994. Vol. 2165 p. 241- 247.

20. TP-Link Archer MR400 інструкція користувача - [Електронний ресурс] Режим доступу - https://www.інструкціїкористувача.com.ua/tp-link/archer-mr400/інструкція-користувача

21. Kibana: Explore, Visualise, Discover data https://www.elastic.co/kibana/

22. Apache Lucene URL: https://lucene.apache.org

23. Snort https://snort.org

24. Scapy https://scapy.net/

25. https://traffic.comics.unina.it/software/ITG/

26. Захист інформації в комп'ютерних системах : підручник для студ. спец. 123 «комп'ютерна інженерія» / уклад. О. М. Гапак, С. І. Балога; рец. : М. І. Глебена. – Ужгород: ПП "АУТДОР-ШАРК, 2021. – 184 с. URL: https://dspace.uzhnu.edu.ua/jspui/handle/lib/36506

27. Комп'ютерні мережі Частина 1 Навчальний посібник [Електронний ресурс]: навч. посіб. для студ. спеціальності 121 «Інженерія програмного забезпечення» та 126 «Інформаційні системи та технології», спеціалізації «Інженерія програмного забезпечення інформаційно управляючих систем» та «Інформаційне забезпечення робототехнічних систем»/ Б. Ю. Жураковський, І.О. Зенів; КПІ ім. Ігоря Сікорського. Київ : КПІ ім. Ігоря Сікорського, 2020. 336 с. URL: https://ela.kpi.ua/handle/123456789/36615

28. Тарнавський Ю. А., Кузьменко І. М. Організація комп'ютерних мереж: підручник: для студ. спеціальності 121 «Інженерія програмного забезпечення» та 122 «Комп'ютерні науки». КПІ ім. Ігоря Сікорського. Київ: КПІ ім. Ігоря Сікорського, 2018. 259 с. URL: https://ela.kpi.ua/handle/123456789/25156

29. Сайко В.Г., Казіміренко В.Я., Літвінов Ю.М. Мережі бездротового широкосмугового доступу. Навчальний посібник. К.: ДУТ, 2015. 196 с. URL: https://dut.edu.ua/uploads/l_881_80314158.pdf

30. Голь В.Д., Ірха М.С. Телекомунікаційні та інформаційні мережі: навчальний посібник. Київ : ІСЗЗІ КПІ ім. Ігоря Сікорського, 2021. 250 с. URL: https://ela.kpi.ua/bitstream/123456789/45409/1/TIM_navch_posib.pdf

31. Eksim Ali. Wireless Communications and Networks - Recent Advances. InTech (March, 2012). 596 p. URL: https://www.intechopen.com/books/1637

32. Bonaventure Olivier. Computer Networking: Principles, Protocols and Practice. Saylor, 2022. 278 p.

33. Sandy Hirtz Education for a Digital World: Advice, Guidelines, and Effective Practice from Around the Globe. 2008. 516 p. URL: http://www.colfinder.org/materials/Education_for_a_Digital_World/Education_for_a_Digital_World_complete.pdf

34. Палеха Ю. І. Етика ділових відносин: Навч. посіб. К.: Кондор, 2007. 356с. URL: https://library.nlu.edu.ua/POLN_TEXT/KNIGI/KONDOR1/

35. CD/ETUKA_DV.pdf

36. Шкіцька І. Ю. Основи академічної доброчесності: практикум: навчально-методичний посібник для студентів вищих навчальних закладів. Тернопіль: ТНЕУ, 2018. 64 с.

37. Computer Engineering Curricula 2016 URL: https://www.acm.org/binaries/content/assets/education/ce2016-final-report.pdf

38. Meticlous Study of Firwall Using Secrity Detection Tools. International Jornal of Computer Aplications & Technology. 2013. Vol.2(1). p. 1-9.

39. McCabe J. Network Analysis, Architecture, and Design. Third edition. / James D. McCabe - Morgan Kaufmann, 2007. 495 p.

40. Mikrotik documentation URL: https://wiki.mikrotik.com.

41. Ileri C.U., Yigit Y., Arapoglu,O., Evcimen H.T., Asci M. Capacitated Graph Theoretical Algorithms for Wireless Sensor Networks Towards Internet of Things. Advances in Wireless Technologies and Telecommunication. 2019.

42. Venetis I.E., Gavalas, D. Pantziou, G., Konstantopoulos C. Mobile agents-based data aggregation in WSNs: benchmarking itinerary planning approaches. Wireless Networks, 24, 2018. p. 2111-2132.

43. “Synthesis of an Expert System for Assessing the Security of Computer Networks Based on a Fuzzy Neural Network.” International Journal of Innovative Technology and Exploring Engineering. 2020.

44. Trushakov, Dmitro et al. “Basic Technical Principleыs Construction of Local Computer Systems for Manaing of Technological Objects.” 2019 IEEE 20th International Conference on Computational Problems of Electrical Engineering (CPEE). 2019. p.1-4.

45. The Cisco Learning Network URL: https://learningnetwork.cisco.com

46. EIA/TIA Standard, Commercial building telecommunications wiring standard (EIA/TIA-568), Electronic Industries Association, Washington, D. C., 1991.

47. Saad, A., Khan, S.A., & Mahmood, A. (2018). A multi-objective evolutionary artificial bee colony algorithm for optimizing network topology design. Swarm Evol. Comput., 38, p.187-201.

48. Методичні рекомендації до виконання кваліфікаційної роботи з освітнього ступеня “Бакалавр” спеціальності 123 «Комп’ютерна інженерія»

галузі знань 12 Інформаційні технології / О.М. Березький, Л.О.Дубчак, Г.М. Мельник, Ю.М. Батько / Під ред. О.М. Березького. Тернопіль: ЗУНУ, 2020. 60с.

49. Методичні вказівки до виконання практичних робіт з дисципліни «Техніко-економічне обґрунтування розробки комп'ютерних систем"/ Н.Я. Савка, І.Р. Паздрій / Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2019. 40 с.

50. Методичні вказівки до оформлення курсових проектів, звітів про проходження практики, випускних кваліфікаційних робіт для студентів спеціальності «Комп'ютерна інженерія» / І.В. Гураль, Л.О. Дубчак / Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2019. 33 с.