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

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2021 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-20М-1,4
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Чаус В.Ю.
«____» _____ 2021 р.

Керівник проекту
кандидат технічних наук
_____ Сергій СМІРНОВ
«____» _____ 2021 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань ._12 _____*"*Інформаційні технології*"*_____

Спеціальність *123 "*Комп'ютерна інженерія*"*

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

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2021 року

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

Чаусу Владиславу Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management*

2. Керівник роботи *Смірнов Сергій Анатолійович, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 42-13 від 02.08.2021 року

3. Строк подання студентом роботи до захисту *10.12.2021 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management*

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

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

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

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2021	14.11.2021
Охорона праці	Оришака О.В.	06.10.2021	16.11.2021

7. Дата видачі завдання « 6 » вересня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2021 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2021 р.	
3.	Розробка моделі компонента	20.10.2021 р.	
4.	Розробка структур даних	25.10.2021 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2021 р.	
6.	Програмування алгоритмів	10.11.2021 р.	
7.	Розрахунок економічної ефективності	13.11.2021 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2021 р.	
9.	Оформлення ПЗ	17.11.2021 р.	
10.	Попередній захист роботи	10.12.2021 р.	

Дата видачі завдання
« 6 » вересня 2021 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2021 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Чаус В.Ю. Дослідження та програмна реалізація системи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

Метою розробки є дослідження та програмна реалізація системи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

Об'єктом дослідження є процес кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

Предметом дослідження є методи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

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

Результат роботи – програмна реалізація системи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

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

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

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

Програму розроблено в середовищі Visual C++.

Ключові слова: комп'ютерна інженерія, Security Information, Event Management

ABSTRACT

Chaus V.Yu. Research and software implementation of cybersecurity system for security audit based on Security Information & Event Management technology. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

In this final qualification work for the second (master's) level of higher education, software has been developed that is designed for a cybersecurity system for security auditing based on Security Information & Event Management technology.

The purpose of the development is research and software implementation of a cybersecurity system for security audit based on Security Information & Event Management technology.

The object of research is the process of cybersecurity for security audit based on Security Information & Event Management technology.

The subject of the research is cybersecurity methods for security audit based on Security Information & Event Management technology.

Research methods are based on methods of information security theory, methods of mathematical statistics, methods of software development.

The result is a software implementation of a cybersecurity system for security audit based on Security Information & Event Management technology.

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

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

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

The program is developed in Visual C ++.

Keywords: computer engineering, Security Information, Event Management

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ІБ	–	інформаційна безпека
ІТ	–	Інформаційні технології
ЛОМ	–	локальна обчислювальна мережа
ММЕ	–	міжмережеві екрани
ATM	–	асинхронний режим передачі
BSD	–	адаптована для Internet реалізація операційної системи UNIX
ICMP	–	міжмережевий протокол управляючих повідомлень
IP	–	Internet Protocol – міжмережевий протокол
NFS	–	мережева файлова система
PPP	–	протокол передачі від точки до точки
RFC	–	опис набору протоколів Internet
RPC	–	віддалений виклик процедури
SIEM	–	Security Information & Event Management
SLIP	–	міжмережевий протокол для послідовного каналу
SMTP	–	Simple Mail Transfer Protocol – простий протокол передачі пошти
TCP	–	Transmission Control Protocol – протокол управління передачею
UDP	–	User Datagram Protocol – протокол користувальницьких датаграм
UNIX	–	багатозадачна операційна система
UTP	–	незахищена вита пара
URL	–	уніфікований покажчик інформаційного ресурсу

ВСТУП

Актуальність теми. Системи захисту постійно розвиваються й адаптуються до нових видів погроз. Кількість джерел інформації, з яких надходять дані по поточному стані захищеності, росте з кожним днем. Коли інфраструктура занадто складна, неможливо встежити за загальною картиною яка відбувається у ній в ній. Якщо вчасно не реагувати на виникаючі погрози й не запобігати їх, користі не буде навіть від сотні систем виявлення вторгнень. На допомогу приходять системи Security Information and Event Management (SIEM).

Перед системою SIEM ставляться наступні завдання.

– Консолідація й зберігання журналів подій від різних джерел – мережевих пристроїв, застосунків, журналів ОС, засобів захисту. Заглянувши в будь-який стандарт ІБ, ви побачите технічні вимоги до збору й аналізу подій. Вони потрібні не тільки для того, щоб виконати вимога стандарту. Бувають ситуації, коли інцидент побачили пізно, а події вже давно затерті або журнали подій чому-або недоступні, і причини інциденту виявити фактично неможливо. Крім того, з'єднання з кожним джерелом і перегляд подій займе багато часу. У противному випадку, без аналізу подій, є ризик довідатися про інцидент у вашій компанії з новиних стрічок.

– Надання інструментів для аналізу подій і розбору інцидентів. Формати подій у різних джерелах розрізняються. Текстовий формат при більших обсягах сильно стомлює, знижує ймовірність виявлення інциденту. Частина продуктів класу SIEM уніфікує події й робить їх більше читабельними, а інтерфейс візуалізує тільки важливі інформаційні події, акцентує на них увагу, дозволяє відфільтрувати некритичні події.

– Кореляція й обробка за правилами. По одній події не завжди можна судити про інцидент. Найпростіший приклад – «login failed»: один випадок нічого не виходить, але три й більше таких події з одним обліковим записом уже можуть

свідчити про спроби підбора. У найпростішому випадку в SIEM правила представлені у форматі RBR (Rule Based Reasoning) і містять набір умов, тригери, лічильники, сценарій дій.

– Автоматичне оповіщення й інцидент-менеджмент. Основне завдання SIEM – не просто зібрати події, але автоматизувати процес виявлення інцидентів з документуванням у власному журналі або зовнішній системі HelpDesk, а також вчасно інформувати про подію.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

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

– Огляд існуючих систем кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

– Дослідження системи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

– Програмна реалізація системи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

Об'єктом дослідження є процес кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

Предметом дослідження є методи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

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

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

– Розроблено вітчизняний продукт кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management

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

Робота апробована на LV Науково-технічна конференція здобувачів вищої освіти «Наука – виробництву», 2021, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №12.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

SIEM здатна виявляти:

– мережеві атаки у внутрішньому й зовнішньому периметрах;

– вірусні епідемії або окремі вірусні зараження, невидалені віруси, бекдори й трояни;

– спроби несанкціонованого доступу до конфіденційної інформації;

– фрод і шахрайство;

– помилки й збої в роботі інформаційних систем;

– уразливості;

– помилки конфігурацій у засобах захисту й інформаційних систем.

Система SIEM універсальна за рахунок своєї логіки. Але для того щоб покладені на неї завдання вирішувалися – необхідні корисні джерела й правила кореляції. Будь-яка подія (наприклад, якщо в певній кімнаті відкрилися двері) можуть бути подані на вхід SIEM і використано.

Джерела вибираються на підставі наступних факторів:

– критичність системи (цінність, ризики) і інформації (оброблюваної й збереженої);

– вірогідність і інформативність джерела подій;

– покриття каналів передачі інформації (повинні враховуватися не тільки зовнішній, але й внутрішній периметр мережі);

– рішення спектру завдань ІТ і ІБ (забезпечення безперервності, розслідування інцидентів, дотримання політик, запобігання витоків інформації й т.п.).

1.2 Область застосування

Основні джерела SIEM:

– Access Control, Authentication – для моніторингу контролю доступу до інформаційних систем і використання привілеїв.

– Журнали подій серверів і робочих станцій – для контролю доступу, забезпечення безперервності, дотримання політик інформаційної безпеки.

– Мережеве активне встаткування (контроль змін і доступ, лічильники мережевого трафіку).

– IDS/IPS. Події про мережеві атаки, зміна конфігурацій і доступ до пристроїв.

– Антивірусний захист. Події про працездатність ПЗ, базах даних, зміні конфігурацій і політик, шкідливих програмах.

– Сканери уразливостей. Інвентаризація активів, сервісів, програмного забезпечення, уразливостей, поставка інвентаризаційних даних і топологічної структури.

– GRC-системи для обліку ризиків, критичності погрози, пріоритезації інциденту.

– Інші системи захисту й контролю політик ІБ: DLP, антифроду, контролю пристроїв і т.п.

– Системи інвентаризації й asset-management. З метою контролю активів в інфраструктурі й виявлення нових.

– Netflow і системи обліку трафіку.

Рішення SIEM містить у собі, як правило, кілька компонентів:

– агенти, установлювані на інспектуєму інформаційну систему (актуально для операційних систем; агент являє собою резидентну програму (сервіс, демон), що локально збирає журнали подій і по можливості передає їх на сервер);

– колектори на агентах, які, по суті, являють собою модулі (бібліотеки) для розуміння конкретного журналу подій або системи;

– сервери-колектори, призначені для попередньої акумуляції подій від безлічі джерел;

– сервер-коррелятор, відповідальний за збір інформації від колекторів і агентів і обробку за правилами й алгоритмами кореляції;

– сервер баз даних і сховища, відповідальний за зберігання журналів подій.

Дані про події збираються від джерел за допомогою встановлених на них агентів, або віддалено (за допомогою з'єднання за протоколами NetBIOS, RPC, TFTP, FTP). У другому випадку неминуче виникає навантаження на мережу й джерело подій, тому що частина систем не дозволяє передати тільки ті події, які ще не були передані, і передає убік SIEM весь журнал подій, що становить найчастіше сотні мегабайт. Затирати ж журнал подій на джерелі при кожному зборі даних – некоректно.

Події повинні не тільки збиратися в консолідоване сховище для розбору по факті інциденту, але й оброблятися. У противному випадку ви одержите рішення, що повністю не виправдує витрати. Безумовно, інструментарій SIEM скоротить час, необхідне для розбору інциденту. Але завдання SIEM – вчасно виявляти, запобігати погрози й оперативно реагувати на них. Для цього необхідно становити правила кореляції – з обліком актуальних для компанії ризиків. Ці правила не перманентні й повинні постійно актуалізуватися експертами. Як і у випадку із правилами для систем виявлення вторгнень, якщо вчасно не прописати правило, що дозволяє виявляти типову погрозу, – вона буде, швидше за все, реалізована. Переваги SIEM перед IDS у правилах – можливість указувати загальний опис симптомів і використання накопиченої статистики baseline для спостереження за відхиленнями від нормального поводження інформаційних систем і трафіку.

Правила віддалено нагадують правила Snort. У них описуються критерії виникнення погрози й реакція на них. Найпростіший типовий приклад з login failed я описував вище. Більше складним прикладом на практиці може бути login

failed у певній інформаційній системі з уточненням групи користувача й ім'я вилученого об'єкта. У випадку із фродом – параметри далекості двох останніх точок використання банківської карти за невеликий інтервал часу (приміром, клієнт скористався оплатою бензину в Кропивницькому, а через 5 мінут де те в Австралії намагаються зняти 5 000 євро).

Реєстрація інцидентів у власній або зовнішній системі HelpDesk відіграє немаловажну роль. По-перше, це документування виникаючих інцидентів. Якщо є зареєстрований інцидент, тобто й відповідальний за його рішення, є строки. Інцидент не залишиться неврахованим (як це буває у випадку оповіщення по електронній пошті). По-друге, це статистика по інцидентах, що дозволяє виявляти проблеми (однотипні інциденти, що повторюються часто й закриваємі без усунення щирих причин). На підставі статистики й розрахунку основних показників можна також судити про ефективність роботи окремих співробітників, підрозділу ІБ, засобів захисту.

За допомогою SIEM можна домогтися майже абсолютної автоматизації процесу виявлення погроз. При коректному впровадженні такої системи підрозділ ІБ переходить на абсолютно новий рівень надання сервісу. SIEM дозволяє акцентувати увагу тільки на критичні й дійсно важливих погрозах, працювати не з подіями, а з інцидентами, вчасно виявляти аномалії й ризики, запобігати фінансові втрати.

Важливо розуміти, що SEIM – це інструмент не тільки ІБ, але й взагалі ІТ. На основі потужних кореляційних механізмів можна ефективно забезпечувати безперервність роботи ІТ-сервісів, виявляти збої в роботі інформаційних і операційних систем, апаратного забезпечення. Крім того, SIEM – інструмент автоматизації. Найпростіший приклад, актуальний для більшості компаній: конфлікт IP-адрес. За рахунок найпростішого правила RBR можна довідатися про інцидент задовго до дзвінка користувача. При цьому усунення причини вимагає набагато менших витрат, а отже, зменшуються можливі фінансові втрати бізнесу.

Якщо аналізувати реальне використання SIEM на практиці, то доводиться визнати, що в більшості випадків робота таких систем спрямована на консолідацію журналів подій від різних джерел. Фактично – використовується тільки функціонал SIM. Якщо і є задані правила кореляції, вони не поповнюються.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Чим гарний безпечник в IT-сфері відрізняється від звичайного? Ні, не тим, що він у будь-який момент часу по пам'яті назве кількість повідомлень, які менеджер Ігор відправила вчора колезі Марії. Гарний безпечник намагається виявити можливі порушення заздалегідь і відловлювати їх у режимі реального часу, додаючи всі сили, щоб не було продовження інциденту. Системи керування подіями безпеки (SIEM, від Security information and event management) значно спрощують завдання швидкої фіксації й блокування будь-яких спроб порушень.

Традиційно SIEM-системи поєднують у собі систему керування інформаційною безпекою й систему керування подіями безпеки. Важливою особливістю систем є аналіз подій безпеки в реальному часі, що дозволяє реагувати на них до настання суттєвого збитку.

Основні завдання SIEM-систем:

- Збір і нормалізація даних.
- Кореляція даних.
- Оповіщення.
- Панелі візуалізації.
- Організація зберігання даних.
- Пошук і аналіз даних.
- Звітність.

Причини високого попиту на SIEM-системи

За останнім часом сильно підвищилися складність і координованість атак на інформаційні системи. Разом з тим ускладнюється й застосовуваний комплекс

засобів захисту інформації – мережеві й хостові системи виявлення вторгнень, DLP-системи, антивірусні системи й міжмережеві екрани, сканери уразливостей та інше. Кожний засіб захисту генерує потік подій з різною деталізацією й найчастіше побачити атаку можна тільки по накладенню подій з різних систем.

Про всілякі комерційні SIEM-системи багато чого написано, але ми пропонуємо короткий огляд безкоштовних повноцінних опенсорсних SIEM-систем, які не мають штучних обмежень на кількість користувачів або обсяги прийнятих/збережених даних, а також легко масштабуються й підтримуються. Сподіваємося, це допоможе оцінити потенціал подібних систем і прийняти рішення, чи варто інтегрувати такі рішення в бізнес-процеси компанії.

AlienVault OSSIM

AlienVault OSSIM – це open-source версія AlienVault USM, однієї з лідируючих комерційних SIEM-систем. OSSIM являє собою фреймворк, що складається з декількох проектів з відкритим вихідним кодом, включаючи мережеву систему виявлення вторгнень Snort, систему моніторингу мереж і вузлів Nagios, хостову систему виявлення вторгнень OSSEC і сканер уразливостей. OpenVAS.

Для моніторингу за пристроями використовується AlienVault Agent, що відправляє журнали з хосту у форматі syslog у платформу GELF або може використовуватися плагін для інтеграції зі сторонніми сервісами, такими як сервіс реверсного проксірування сайтів Cloudflare або системою багатофакторної автентифікації Okta.

Версія USM відрізняється від OSSIM розширеною функціональністю керування журналами, моніторингу хмарної інфраструктури, автоматизації, і обновлюваною інформацією про погрози й візуалізацію.

Переваги

– Побудована на перевірених open-source проектах.

– Велике співтовариство користувачів і розроблювачів.

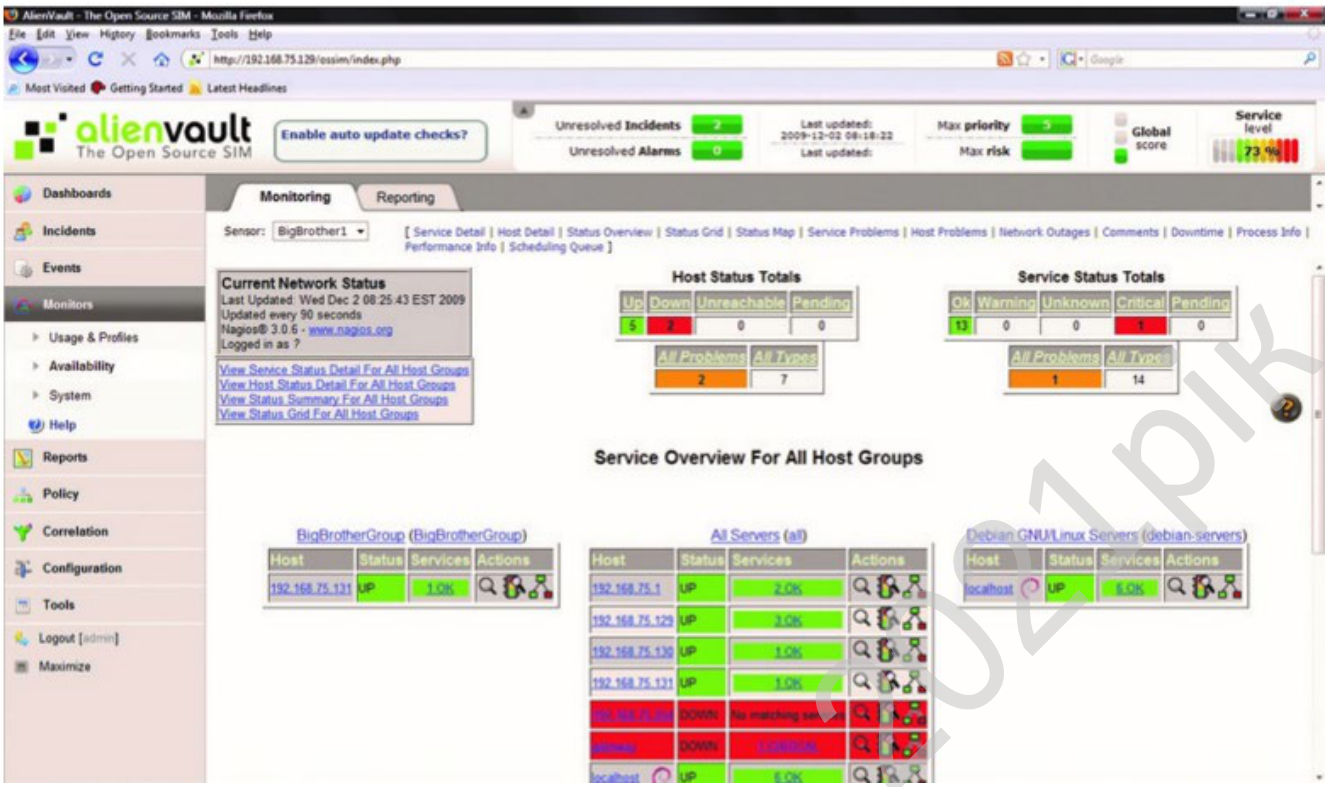

Рисунок 2.1 – Інтерфейс користувача AlienVault Agent

Недоліки

– Не підтримує моніторинг хмарних платформ (наприклад, AWS або Azure).

– Відсутнє керування логми, візуалізація, автоматизація й інтеграція зі сторонніми сервісами.

MozDef (Mozilla Defense Platform)

Розроблена Mozilla SIEM-система MozDef використовується для автоматизації процесів обробки інцидентів безпеки. Система розроблена з нуля для одержання максимальної швидкодії, масштабованості й відказостійкості, з мікросервісною архітектурою – кожний сервіс працює в контейнері Docker.

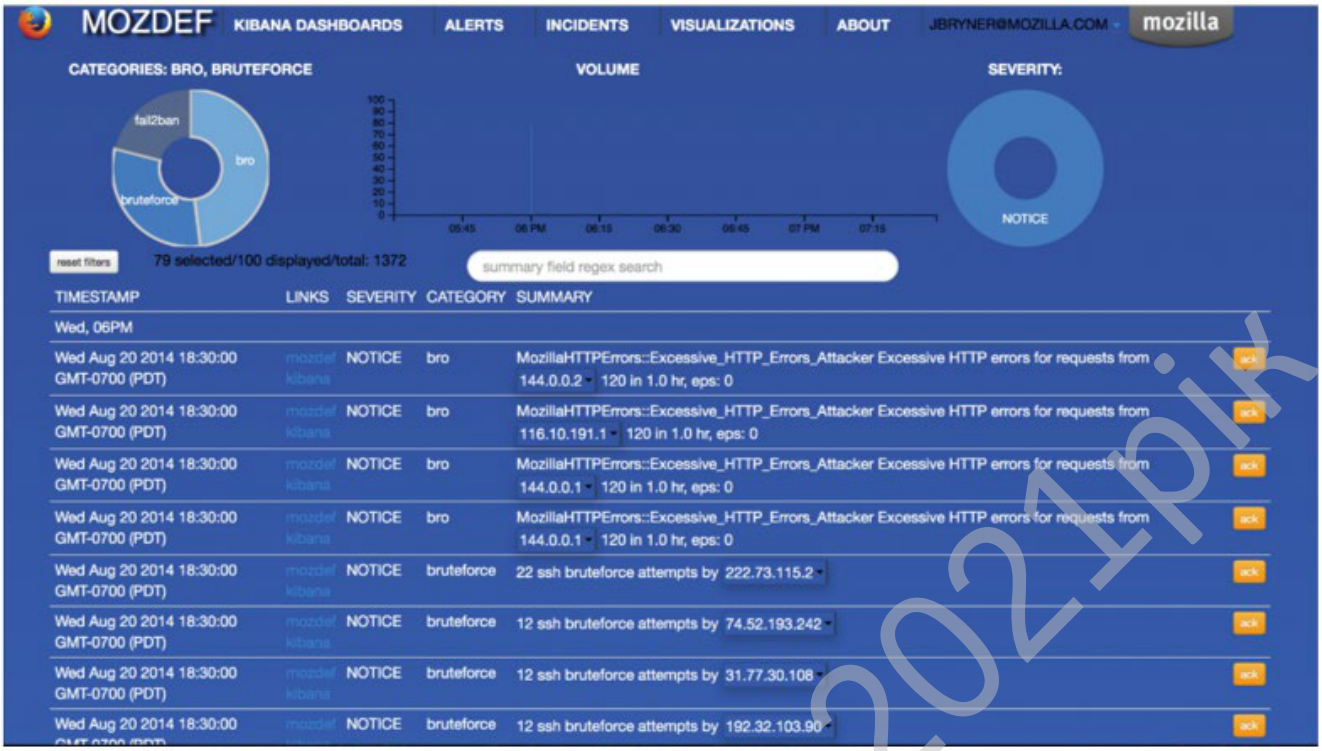

Рисунок 2.2 – Інтерфейс користувача MozDef

Як і OSSIM, MozDef побудована на перевірених часом опенсорсних проектах, що включають модуль індексування логів і пошуку Elasticsearch, платформу Meteor для побудови гнучкого web-інтерфейсу, і плагін Kibana для візуалізації й побудови графіків.

Кореляція подій і оповіщення виконується з використанням запитом Elasticsearch, що дозволяє написати власні правила обробки подій і оповіщення з використанням Python. Зі слів Mozilla, MozDef може обробляти більше 300 мільйонів подій у день. MozDef приймає події тільки у форматі JSON, але є інтеграція зі сторонніми сервісами.

Переваги

– Не використовує агентів – працює зі стандартними логми JSON.

– Легко масштабується завдяки мікросервісній архітектурі.

– Підтримує джерела даних хмарних сервісів, включаючи AWS CloudTrail і GuardDuty.

Недоліки

– Нова й менш устояна система.

Wazuh

Wazuh зпочатку розвивалася як форк OSSEC, однієї із самих популярних SIEM з відкритим кодом. І тепер це власне унікальне рішення з новою функціональністю, виправленими помилками й оптимізованою архітектурою.

Система побудована на стеці ElasticStack (Elasticsearch, Logstash, Kibana) і підтримує як збір даних на основі агентів, так і прийом системних журналів. Це робить її ефективної для моніторингу пристроїв, які генерують журнали, але не підтримують установку агента – мережеві пристрої, принтери й периферія.

Wazuh підтримують існуючі агенти OSSEC і навіть полишає керівництво по міграції з OSSEC на Wazuh. Хоча OSSEC усе ще активно підтримується, Wazuh розглядається як продовження OSSEC через додавання нового веб-інтерфейсу, REST API, більше повного набору правил і багатьох інших поліпшень.

Переваги

– Засновано й сумісна з популярною SIEM OSSEC.

– Підтримує різні варіанти установки: Docker, Puppet, Chef, Ansible.

– Підтримує моніторинг хмарних сервісів, включаючи AWS і Azure.

– Включає комплексний набір правил, для виявлення безлічі типів атак і дозволяє зіставляти їх у відповідність із PCI DSS v3.1 і CIS.

– Інтегрується із системою зберігання й аналізу логів Splunk візуалізації подій і підтримки API.

Недоліки

– Складна архітектура – вимагає повного розгортання Elastic Stack на додаток до серверних компонентів Wazuh.

Prelude OSS

Prelude OSS – це open-source версія комерційної Prelude SIEM, розробленою французькою компанією CS. Рішення являє собою гнучку модульну

SIEM-систему, що підтримує безліч форматів логів, інтеграцію зі сторонніми інструментами такими як OSSEC, Snort і мережеву систему виявлення Suricata.

Кожна подія нормалізується в повідомлення по форматі IDMEF, що спрощує обмін даними з іншими системами. Але є й ложка дьогтю – Prelude OSS сильно обмежена по продуктивності й функціональності в порівнянні з комерційною версією Prelude SIEM, і призначена скоріше для невеликих проектів або для вивчення рішень SIEM і оцінки Prelude SIEM.

Переваги

– Випробувана часом система, розроблювальна з 1998 р..

– Підтримує безліч різних форматів логів.

– Нормалізує дані до формату IMDEF, що дозволяє легко передавати дані в інші системи безпеки.

Недоліки

– Значно обмежена по функціональності й продуктивності в порівнянні з іншими open-source SIEM-системами.

Sagan

Sagan – це високопродуктивна SIEM, що підкреслює сумісність із Snort. Крім підтримки правил, написаних для Snort, Sagan може виконувати запис у базу даних Snort і навіть може використовуватися з інтерфейсом Shuil. По суті, це легке багатопоточне рішення, що пропонує нові функції, залишаючись дружнім до користувачів Snort.

Переваги

– Повністю сумісна з базою даних Snort, правилами, і користувальницьким інтерфейсом.

– Багатопоточна архітектура забезпечує високу продуктивність.

Недоліки

– Порівняно молодий проект із невеликим співтовариством.

– Складний процес установки, що включає складання всієї SIEM з вихідних кодів.

Висновок

У кожної з описаних SIEM-систем є свої особливості й обмеження, тому їх не можна назвати універсальним рішенням для будь-якої організації. Однак у цих рішень відкритий код, що дозволяє розгортати, тестувати і оцінювати їх без надмірних витрат.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для реалізації програми мною була використана мова програмування Visual C++. У зв'язку з тим, що сьогодні рівень складності програмного забезпечення дуже високий, розробка застосунків Windows з використанням тільки якої-небудь мови програмування значно утрудняється. Програміст повинен затратити масу часу на рішення стандартних завдань по створенню багатовіконного інтерфейсу. Реалізація технології зв'язування й вбудовування об'єктів – OLE – зажадає від програміста ще більш складної роботи. Щоб полегшити роботу програміста практично всі сучасні компілятори з мови C++ містять спеціальні бібліотеки класів. Такі бібліотеки містять у собі практично весь програмний інтерфейс Windows і дозволяють користуватися при програмуванні засобами більш високого рівня, чим звичайні виклики функцій. За рахунок цього значно спрощується розробка застосунків, що мають складний інтерфейс користувача, полегшується підтримка технології OLE і взаємодія з базами даних. Сучасні інтегровані засоби розробки застосунків Windows дозволяють автоматизувати процес створення застосунка. Для цього використовуються генератори застосунків. Програміст відповідає на питання генератора застосунків і визначає властивості застосунка – чи підтримує воно багатовіконний режим, технологію OLE, тривимірні органи керування, довідкову систему. Генератор застосунків, створить додаток, що відповідає вимогам, і надасть вихідні тексти. Користуючись їм як шаблоном, програміст зможе швидко

розробляти свої застосунки. Подібні засоби автоматизованого створення застосунків включені в компілятор Microsoft Visual C++ і називаються MFC AppWizard. Заповнивши кілька діалогових панелей, можна вказати характеристики застосунка й одержати його тексти, постачені великими коментарями. MFC AppWizard дозволяє створювати одновіконні й багатовіконні застосунки, а також застосунки, що не мають головного вікна, – замість нього використовується діалогова панель. Можна також включити підтримку технології OLE, баз даних, довідкової системи. Звичайно, MFC AppWizard не всесильний. Прикладну частину застосунка програмістові прийдеться розробляти самостійно. Вихідний текст застосунка, створений MFC AppWizard, стане тільки основою, до якої потрібно підключити інше. Але працюючий шаблон застосунка – це вже половина всієї роботи. Вихідні тексти застосунків, автоматично отриманих від MFC AppWizard, можуть становити сотні рядків тексту. Набір його вручну був би дуже стомлюючий. Потрібно відзначити, що MFC AppWizard створює тексти застосунків тільки з використанням бібліотеки класів MFC (Microsoft Foundation Class library). Тому тільки вивчивши мову C++ і бібліотеку MFC, можна користуватися засобами автоматизованої розробки й створювати свої застосунки в найкоротший термін. Як уже згадувався, MFC – це базовий набір (бібліотека) класів, написаних мовою C++ і призначених для спрощення й прискорення процесу програмування для Windows. Бібліотека містить багаторівневу ієрархію класів, що нараховує близько 200 членів. Вони дають можливість створювати Windows-застосунки на базі об’єктно-орієнтованого підходу. З погляду програміста, MFC являє собою каркас, на основі якого можна писати програми для Windows. Бібліотека MFC розроблялася для спрощення завдань, що стоять перед програмістом. Як відомо, традиційний метод програмування під Windows вимагає написання досить довгих і складних програм, що мають ряд специфічних особливостей. Зокрема, для створення тільки каркаса програми таким методом знадобиться близько 75 рядків коду. У міру ж збільшення складності програми її код може досягати воістину неймовірних розмірів. Однак та ж сама програма,

написана з використанням MFC, буде приблизно в три рази менше, оскільки більшість приватних деталей приховано від програміста.

Одною з основних переваг роботи з MFC є можливість багаторазового використання того самого коду. В зв'язку з тим, що бібліотека містить багато елементів, загальних для всіх Windows-застосунків, немає необхідності щораз писати їх заново. Замість цього їх можна просто успадковувати (говорячи мовою об'єктно-орієнтованного програмування). Крім того, інтерфейс, забезпечуваний бібліотекою, практично незалежний від конкретних деталей, його що реалізують. Тому програми, написані на основі MFC, можуть бути легко адаптовані до нових версій Windows (на відміну від більшості програм, написаних звичайними методами). Ще однією істотною перевагою MFC є спрощення взаємодії із прикладним програмним інтерфейсом (API) Windows. Будь-який додаток взаємодіє з Windows через API, що містить кілька со тень функцій. Значний розмір API утрудняє спроби зрозуміти й вивчити його цілком. Найчастіше навіть складно простежити, як окремі частини API зв'язані один з одним! Але оскільки бібліотека MFC поєднує (шляхом інкапсуляції) функції API у логічно організовану безліч класів, інтерфейсом стає значно легше управляти.

Оскільки MFC являє собою набір класів, написаних мовою C++, тому програми, написані з використанням MFC, повинна бути в той же час програмами на C++. Для цього необхідно володіти відповідними знаннями. Для початку необхідно вміти створювати власні класи, розуміти принципи спадкування й уміти перевизначати віртуальні функції. Хоча програми, що використовують бібліотеку MFC, звичайно не містять занадто специфічних елементів з арсеналу C++, для їхнього написання проте потрібні солідні знання в даній області.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне

забезпечення, яке призначено для системи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

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

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

З появою перших засобів захисту інформації виникли перші пекучі питання: як довідатися, що зведені барикади працюють і захищають? як швидше реагувати на оповіщення? як зрозуміти, які погрози вдалося запобігти? Чи працює наш файерволл, можна довідатися, виконавши ICMP ping: якщо правила в ACL (access control list) працюють, то відповідей, що містять echo reply, бути не повинне. Можна через консоль пристрою переглянути журнал подій, розбираючи сотні або тисячі рядків вручну й намагаючись побачити відбиту або виявлену погрозу.

Час – гроші

Журналів подій, одержуваних від одних тільки активних засобів захисту, – багато, не говорячи вже про критичні сервери, бази даних, застосунках. За допомогою цих журналів можна виявити несанкціоновані спроби доступу, мережеві атаки, аномалії, що ведуть до порушення безперервності бізнесу або політик безпеки. Щоб відкрити журнал подій, потрібно виконати послідовність дій, які вимагають часу: запустити додаток, підключитися до консолі, вивести список подій і вивчити його. Навіть якщо допустити, що один співробітник відповідає тільки за контроль антивірусного ПЗ (припустимо, що є централізована консоль керування), установку відновлень і IPS (припустимо, що їх не більше 2-4), – на перегляд подій від цих джерел і розбір проблем за останню добу в нього піде біля години. Відзначимо людський фактор: офіце р може бути завантажений іншими справами, може боліти або бути у відпустці, відволіктися від роботи або виконати її для проформи. Тепер порахуйте, скільки людино-годин потрібно, щоб аналізувати журнали подій на критичних активах хоча б раз у добу? Урахуйте в розрахунках заробітну плату кваліфікованого співробітника,

здатного виявити погрози в цих журналах подій, урахуйте час, необхідне для підключення до СЗІ у вашій філії по повільних каналах зв'язку. Дорого? Так, виходить кругла сума, назвавши яку керівництву, ви, швидше за все, будете виставлені з кабінету.

Отже, ви встановили засоби захисту, настроїли їх, вони працюють, – що вам ще потрібно? SIEM заміняє не один десяток людей, працює оперативно й не буде просити про підвищення зарплати.

Захист бізнесу

Головне завдання ІБ – забезпечити захист бізнесу й безперервність бізнес-процесів. Що для цього потрібно? Описуються бізнес-процеси, визначаються активи, проводиться їхній аудит (включаючи сканування й пентести), складається модель порушника, вивчаються ризики, складається план їхньої мінімізації. Які міри приймаються для мінімізації ризику? Створюються політики, проводиться навчання користувачів, установлюються засоби захисту інформації, змінюються конфігурації, установлюються відновлення…Ми все це зробили – і залишили як є? до наступного циклу PDCA?

Тримати руку на пульсі

Принцип «поставити й забути» в ІБ не застосуємо. Абсолютного захисту не існує, і самі малоймовірні ризики можуть гукнутися зупинкою бізнесу й величезних фінансових втрат. Будь-яке програмне й апаратне забезпечення може перестати працювати або бути невірно налаштоване – і пропустити погрозу. Бачили панель керування в сучасних літаках? Всі важливі індикатори зібрані воєдино з дотриманням ергономіки й пріоритету. Пілот і його помічник не можуть не побачити порушення критично важливого показника. Так і в SIEM: у випадку будь-якого відхилення від baseline або політик (курс літака) або порушення працездатності активу в результаті збоїв або погроз (неполадки в устаткуванні) – оператори-пілоти будуть негайно сповіщені.

Чому негайно й що буде через годину? Віруси поширюються в лічені секунди, у зловмисників є автоматизовані комплекси для аналізу й експлуатації

уразливостей. Подія про що посипались RAID-масиви в системі, що перебуває в промисловій експлуатації, завтра вам буде вже не цікаво, тому що частина даних (або навіть усі) будуть загублені. Чим більш оперативно ви будете інформовані, чим швидше уживете заходів, – тим менше фінансових втрат понесе ваш бізнес. Добре, якщо той що відбувся інцидент не буде мати наслідків.

Превентивного захисту не існує

Якщо ми встановлюємо централізоване антивірусне ПЗ – необхідно переконатися, що воно встановлено скрізь, правильно зконфігуровано, працює з актуальними базами. Як? За допомогою журналів подій.

Навіщо? Представте, що ви автоматизували установку корпоративного антивірусного ПЗ й відновлення баз. Аудит журналу подій ви здійснюєте раз у два-три дні, але відбувся збій, в ОС на робочій станції, що коштує на складі, не запускається сервіс і вона заражена вірусом, що поширюється по мережі. Абсолютно не факт, що на всіх серверах заборонений, приміром, автозапуск, установлені всі заплатки: у реальному житті такого практично не буває. Автоматично розміщений використовуючий уразливість троян з автозапуском і поширенням по мережі або підробленому ярлику на загальному мережевому ресурсі приводить до колапсу всієї компанії. Поки ви будете в авральному режимі аналізувати те, що відбулося – бізнес, швидше за все, буде простоювати, а начальство – нервувати й обурюватися. Фінансову оцінку збитків від простою підприємства легко зробити самостійно, благо це не так складно. Крім того, подібні збої мають властивість негативно впливати на премії й зарплату.

Контрольована погроза, прийнятий ризик

На практиці зустрічаються випадки, коли безпека йде врозріз із бізнесом. Трапляються ситуації, коли не можна поставити відновлення, щоб закрити уразливість (причин маса: сертифікація, нестабільність роботи, «не протестовано», конфлікт із іншим ПЗ), або, наприклад, неможливо заборонити RPC, оскільки бізнес-додаток перестане працювати. Витрати на усунення погрози можуть перевищувати можливі втрати, тому ризик «приймається». Однак ми

можемо контролювати подібні ризики за допомогою SIEM, реагувати на виникаючі інциденти, повертаючи по закінченні року засоби, виділені на покриття операційних ризиків, назад у бюджет. Природно, що в цьому випадку не може бути й мови про перегляд оператором журналів міжмережевого екрана без автоматичного аналізу й реєстрації інцидентів як способу контролю ризиків.

Немає причини – усі винуваті

Ви напевно зіштовхувалися з випадками, коли для рішення інциденту немає даних: відсутня інформація про точний час і місце виникнення (не вважаємо дзвінки від користувачів), про те, що передувало інциденту; і ми не можемо відповісти на головні питання – чому відбувся інцидент і хто в цьому винуватий. Ні, це потрібно не для того, щоб покарати винних (хоча це теж іноді необхідно). Головне, що необхідно з'ясувати за підсумками інциденту, – що робити, щоб інцидент не повторився. Причому одного тільки убудованого в ОС журналу (Windows event log або syslog) може виявитися недостатньо.

Системний адміністратор

У зрілій розгалуженій інфраструктурі права адміністрування делегуються досить широкому колу співробітників. Природно, всі ці співробітники проходять перевірку служби безпеки, і ми їм довіряємо. Але на практиці найчастіше позначається людська психологія: співробітник, порушивший базу даних, RAID, що приніс на особистій флешці вірус, через якого «устав» бізнес-процес, під страхом звільнення й штрафу, загнаний у кут – заметає сліди, видаляючи або підробляючи журнали подій. Якщо не зібрати вчасно ці журнали, то бізнесу буде нанесена шкода у вигляді фінансових втрат і зіпсованої репутації. Зібрані вчасно й консолідовані в сховище журнали подій допоможуть вам прийняти правильне рішення за підсумками інциденту. Здійснити видалення даних (подій і інцидентів) з SIEM непомітно не вийде: залишаються записи в системному журналі, здійснюється контроль цілісності. Докази у вигляді журналів подій в SIEM-системах допоможуть вашої організації в рішенні судових питань.

Хто знає, що це за скрипт?

Безумовно, можна побудувати керування журналами і яке-ніяке керування подіями на «самописних» сценаріях. Журнали збирати через syslog або відкрите ПЗ. Можна все оформити на PowerShell, «батники», sh-сценарії, а про інциденти повідомляти на електронну пошту. Як зручно й дешево!

Так, це прийнятно для малого бізнесу. Повернемося до нашого приклада з літаком. Заберемо подумки всі індикатори із приладової панелі (або зітремо їхньої назви), а повідомлення про неполадки будемо направляти пілотові по СМС і на електронну пошту…Як швидко пілотові набридне лазить у кишеню за телефоном і розбирати вхідні листи?

SIEM-системи мають функцію самодіагностики й контролю роботи компонентів. Це не розкидані отут і там «батники», цілісність і працездатність яких дуже важко проконтролювати. При використанні розрізнених сценаріїв практично неможливо буде захиститися від підміни вмісту або перегляду адміністративного облікового запису в незашифрованому виді. На відміну від SIEM: це комплексна система, що повідомляє про безперервність збору подій, про збої в роботі своїх компонентів, про доступ до системних функцій і т.п.

Захищайте не тільки критичні активи

Представимо, що ви захистили критичні (на вашу думку) активи, наприклад бізнес-додаток або базу даних. Все добре, грошей витрачено в міру, заощадили на відсутності СЗІ для робочих станцій і двофакторної авторизації для мобільних користувачів. Користувачів «затисли» груповими політиками. От тільки не врахували, що двері із замком, що коштує посередині поля, – абсолютно неефективна. Зловмисники одержать користувальницький і адміністративний аккаунт із незахищених робочих станцій або з мобільних пристроїв і з абсолютно легітимними запитами до вашої суперзахищеної бази даних «витягнуть» усе, що тільки можна. Деструктивні дії давно не в моді. Ви довідаєтеся про витік інформації з новин – і зачудуєтеся: адже всі ваші сервери були надійно захищені! Це приклад типової атаки по моделі APT. Запущені

процеси, нові бібліотеки в ОС, нові сервіси, відкриті порти й з'єднання, підвищення привілеїв – все це можна побачити в журналах подій на робочих станціях, які не були, на вашу думку, критичними активами…

Захист повинна бути комплексним. Доказ тому – інциденти з Bit9 і RSA, які чомусь не поставили розроблювальний ними захист на свої ж робочі станції.

Подання

Засоби захисту, як правило, є сигнатурними, тобто створюються на основі аналізу вже відомих погроз (віруси, мережеві атаки, навіть словники в DLP). Нові погрози ви можете виявити тільки із застосуванням складних алгоритмів кореляції, на основі мільйонів подій і показників, а також аналізу baseline. Людський мозок не завжди здатний комплексно проаналізувати такий обсяг даних. Однак абстракція подань в SIEM-системах сприяє своєчасному виявленню погроз операторами. Система робить всі попередні розрахунки й виводить показники. Як мінімум, приміром, на основі аналізу baseline система повідомляє про новий DynDNS-трафік, про те, що зафіксовано по 10 безуспішних спроб входу з різних активів від імені доменного адміністратора. Як правило, система здатна повідомити про троян або брутфорс (залежно від складу правил кореляції й можливостей конкретної системи). Застосування більше складних алгоритмів кореляції дозволить довідатися причину інциденту (наприклад, виявити підключення користувачем модему, у результаті якого відбулося зараження трояном і брутфорс). Людині не під силу самостійно здійснювати такий аналіз на підставі мільйонів текстових подій. Можливість налаштовування панелей візуалізації корисна як окремим співробітникам, так і для роботи SOC (security operation center), а також підрозділів IT і техпідтримки.

Відповідність (compliance)

У ряді регіональних, міжнародних, національних, галузевих стандартів є вимоги до організації процесу керування журналами. Всі SIEM-системи мають шаблони, що відповідають міжнародним стандартам, і можливість додавання своїх шаблонів для формування звіту про відповідність до збору й зберіганню

подій. У випадку із саморобною системою вам доведеться витратити значні ресурси, щоб зробити подібні шаблони у форматі звітів або інтерфейсу для аудитора.

Акценти

Невірне реагування на інциденти порівнянно з некоректним поводженням світлофора. ІБ- і ІТ-підрозділи будуть нездатні вирішувати першочергові завдання по забезпеченню бізнес-процесів. SIEM має мінімально необхідні засоби організації процесу реєстрації інцидентів (або має можливість інтеграції зі службою підтримки), що сприяє контролю рішення інцидентів і нагромадження бази знань. В SIEM є можливість інтеграції й пріоритезації інцидентів залежно від їхнього впливу на бізнес -процеси, від цінності активу й небезпеки погрози. У деяких системах можлива інтеграція із системами керування ризиками.

Існує помилкова думка, що SIEM плодить велику кількість інцидентів, на які підрозділ ІБ попросту не встигає реагувати. Необхідно розуміти, що SIEM – це не рішення «з коробки» і, як і у випадку з DLP -системами, тут необхідно правильне впровадження, інтеграція із джерелами подій, індивідуальний підхід до активного набору правил і алгоритмам кореляції. Гнучка система виключень і правильне налаштовування SIEM гарантують вам акцент тільки на критично важливих подіях – без флуда.

Поділитеся подіями

SIEM – це система не тільки для ІБ. Помилки й збої в операц ійних системах, мережевому встаткуванні, ПЗ – інформацію про усе це співробітники ІТ-відділу можуть почерпнути в SIEM. ІТ-відділу також хочеться дізнаватися про виникаючі інциденти не по дзвінку користувачів, а заздалегідь (тим більше, що – як і інциденти ІБ – ІТ-інциденти можна запобігти).

SIEM – не занадто просте рішення для процесу керування журналами, до того ж досить дороге для впровадження в малому й середньому бізнесі. Для його експлуатації вам необхідно мати як мінімум одного кваліфікованого співробітника, що буде забезпечувати контроль безперервності збору подій,

управляти правилами кореляції, коректувати й обновляти їх з появою нових погроз і відповідно до змін в інфраструктурі. Установка SIEM у якості «чорного ящика» з активацією всіх передвстановлених правил кореляції без належного контролю й керування приведе до розтрати бюджету.

При успішному впровадженні ви одержите:

– кореляцію й оцінку впливу IT- і ІБ-подій і процесів на бізнес;

– SOC з аналізом ситуації в інфраструктурі в режимі реального часу;

– автоматизацію процесів виявлення погроз і аномалій;

– автоматизацію процесів реєстрації й контролю інцидентів;

– аудит політик і стандартів відповідності, контроль і звітність;

– задокументоване коректне реагування на виникаючі погрози ІБ і IT у режимі реального часу із пріоритезацією залежно від впливу погроз на бізнес-процеси;

– можливість розслідування інцидентів і аномалій, у тому числі ті, що відбулись давно;

– доказову базу для судових розглядів;

– звітність і показники (KPI, ROI, керування подіями, керування уразливостями).

Я навів лише деякі приклади того, як SIEM допоможе вашому бізнесу в забезпеченні безперервності, підвищенні оперативності, у рішенні проблем і інцидентів.

3.2 Розробка структурної схеми

Давайте подивимося, із чого структурно складається SIEM. Як видно, для SIEM характерно більше число джерел і вони одержали можливість реагувати на «позаштатні» ситуації: наприклад, якщо в користувача раптово помінялася активність (раніше просто переглядав сторінки, використовуючи HTTP, а зараз починає активно ганяти трафік «назовні» через інші протоколи, наприклад) – те

це привід згенерувати «подію». Крім цього, SIEM здатні аналізувати минаючий у мережі трафік без використання додаткового «заліза» або ПЗ (шляхом перекладу вільної мережевої карти сервера SIEM в «нерозбірливий» режим роботи), відслідковувати активність застосунків – і це крім основної функції, що залишилася, «збір логів із джерел» і «аналіз подій». Також SIEM можуть відслідковувати й віртуальні інфраструктури.

Рисунок 3.1 – Структурна схема системи

Та й сам аналіз став більше інтелектуальним – число «злочинів», що вимагають розгляди фахівцем, при використанні SIEM зменшується десь на 25-30%, це досягається за рахунок того, що вже накопичено якусь статистику рутинних операцій, установлюється якийсь поріг спрацьовування для «нетипових» операцій (скажемо, користувач user1 звичайно займається редагуванням і відправленням документів по smtp, але якщо він для відправлення раптом стане використовувати інший протокол – та подія, що вимагає уваги, буде згенеровано не відразу, а тільки при подоланні деякого порога частоти (або кількості) таких подій. Зрозуміло, ураховуються дані мережевої активності й т.д.).

Але! Багато менеджерів затверджують, що якщо у вас є SIEM %siem_name%, те немає необхідності установки DLP, IDS, сканерів уразливостей

і т.д. Насправді, це не так. SIEM може відстежити якісь аномалії в мережевому потоці, але нормальний аналіз вона провести не зможе. SIEM, власне кажучи, марна без інших систем безпеки. Основна перевага SIEM – збір, зберігання й аналіз логів – буде зменшено на 0 без джерел цих самих логів.

Часта фраза в листівках «SIEM %siem_name% – легкість і швидкість впровадження, мінімум помилкових спрацьовувань, не вимагає переналаштовування вже наявних засобів безпеки...». Це не так. Можна, звичайно, обійтися мінімумом переналаштовувань – просто перенаправляти весь потік подій із пристроїв і систем безпеки на SIEM. На правильно налаштованій SIEM це не викличе значного збільшення числа помилкових спрацьовувань, але серйозно навантажить сервер БД (і викличе розростання БД). Що в підсумку виллється в збільшенні часу обслуговування БД і, можливо, до пропуску якихось дійсно важливих інцидентів безпеки. Тому, подумати – що ж саме відправляти на SIEM із уже наявних пристроїв, а що залишити на відкуп уже наявної системи безпеки – прийде. Приклад – можна перенаправляти всі логи антивірусу, установленого на ПК користувача, безпосередньо на SIEM – включаючи події про відновлення баз – але потрібно чи це вам? Особливо якщо врахувати, що деякі джерела можуть генерувати однотипні або повторювані події. Звичайно, в SIEM у більшості випадків передбачена можливість їхнього об'єднання, але це знову-же викликає зайві навантаження на сервер SIEM.

Варто особливо звернути увагу на можливість збирати flow-потоки (NetFlow, sFlow і т.д.) – корисна річ для відсічення зайвої інформації про трафік в мережі й, у той же час, одержання додаткової корисної інформації про стан мережі, отриманої безпосередньо з мережевих пристроїв.

Також як не буває легкого впровадження. Перед запуском потрібно провести масу аналітичної роботи, визначити, які події важливі, які немає й т.д.

В остаточному підсумку – питання – кому потрібні системи такого класу? Потрібні ці системи, на погляд автора, тим, у кого більша мережева інфраструктура й хто хоче хоч якось упорядкувати події й бути в курсі

інцидентів. Але при цьому готовий до більших витрат, окупність таких систем не моментальна, користь, на перший погляд, не очевидна. До того ж, дані системи вимогливі до «заліза». Хоча, була в одній листівці вдала фраза: «SIEM дозволяє CIO пояснити проблеми IT мовою бізнесу». До того ж, дані системи вимогливі до «заліза».

Фраза не позбавлена підстав: звіти, створювані сучасними SIEM, мало того, що в різних форматах, так ще й, що налаштовуються під потреби конкретної організації, найчастіше дозволяють одержати всі необхідні дані на двох-трьох аркушах формату А4, представлені у вигляді зрозумілих і наочних графіків або цифр.

На закінчення, хочеться підбити підсумок: SIEM (або продуктів, що називають себе такими) – багато. Але, оскільки по кишені вони в основному лише великим замовникам – вони будуть звертати увагу на лідерів.

У світлі цього, не зовсім ясно, які перспективи в опенсорсних SIEM-систем, на мій погляд, це не той продукт, що легко можна замінити опенсорсним без ризику втрати переваг у вигляді регулярних відновлень і кваліфікованої підтримки. З іншого боку, існує позитивний приклад OpenBSD.

3.3 Розробка функціональної схеми

Абревіатура SIEM утворена від Security Information & Event Management, що дослівно можна перевести як система керування подіями й інформаційною безпекою. SIEM забезпечує аналіз у реальному часі подій, що відбуваються в IT-інфраструктурі. Подібний аналіз необхідний для виявлення й визначення серед всіх подій інформаційної безпеки й реагування на них.

Розглянемо з яких функціональних блоків складається система.

Рисунок 3.2 – Функціональна схема системи

Блок функціоналу SIEM систем

SIEM системи, поза залежністю від виробника, мають наступний функціонал:

– Агрегація даних – збір, обробка й зберігання логів з різних пристроїв і застосунків.

– Кореляція подій – пошук загальних атрибутів події. Подібна технологія забезпечує застосування різних технічних прийомів для інтеграції даних з різних джерел для перетворення вихідних даних в інформацію з якої можна працювати далі.

– Оповіщення – SIEM системи підтримують можливість оповіщення про події за різними каналах зв'язку.

– Аналіз і керування ризиками безпеки.

– Проведення розслідування інцидентів.

– Формування звітів.

– Реакція на атаки.

Блок джерел даних для SIEM

Джерелами даних для SIEM систем є:

– IDS/IPS системи.

– Антивірусні програми.

– Журнали подій операційних систем.

– Міжмережеві екрани.

– Сканери уразливостей.

– Системи інвентаризації.

– Проксі-сервера.

– Системи автентифікації.

Блок виявлення проблем з кібербезпекою

SIEM-рішення дозволяє виявити:

– Зовнішні й внутрішні кібератаки,

– Окремі зараження й вірусні епідемії.

– Спроби одержати несанкціонований доступ до захищених інформаційних потоків.

– Факти корпоративного шахрайства.

– Погрішності й порушення в роботі інформаційних систем.

– Слабкі точки захисту.

– Порушення структури засобів захисту.

– Цільові розкрадання.

Блок типових сценаріїв використання SIEM-системи

Розглянемо популярні сценарії використання SIEM:

– Відстеження автентифікації й виявлення компрометації аккаунтів користувачів і адміністраторів.

– Відстеження випадків зараження. Виявлення шкідливих програм з використанням журналів вихідних паперів брандмауера й журналів веб-проксі, а також внутрішніх журналів підключення й мережевих потоків.

– Моніторинг підозрілого вихідного трафіку й переданих по мережі даних з використанням журналів брандмауера, журналів веб-проксі й NetFlow. Виявлення крадіжки даних і інших підозрілих зовнішніх з'єднань.

– Відстеження системних змін і інших адміністративних дій у внутрішніх системах і їхньої відповідності дозволеній політиці.

– Відстеження атак на веб-застосунки і їх наслідків з використанням журналів веб-сервера, WAF (Web Application Firewall, екран для захисту веб-застосунків) і логів застосунків. Виявлення спроб компрометації веб-застосунків шляхом аналізу різних звітів.

Деякі компанії задаються питанням, чи актуальні SIEM-системи або ж цей підхід уже застаріло.

Для відповіді на це питання необхідно зрозуміти, як варто використовувати SIEM.

SIEM не буде протидіяти хакерам, це рішення для моніторингу, що вдосконалюється в міру розвитку технології.

Більше того, без SIEM неможливо побудувати такі системи й центри моніторингу й реагування, як SOC (Security Operation Center), тому що SIEM допомагає вирішувати цілий ряд ключових завдань: збирати й зберігати лог-файли в єдиному централізованому сховищі, надавати спеціалізовані звіти

аудиторам для відповідності вимогам законодавства й галузевих стандартів і виконувати кореляцію подій між різними джерелами.

Варто приділити особливу увагу налаштовуванню SIEM під клієнта, його інфраструктуру й системи безпеки.

Добре налаштовані правила кореляції дозволять операторові аналізувати дійсно важливі повідомлення про інциденти, відсіваючи зайве.

Як вибрати SIEM-рішення?

Системи SIEM, представлені на ринку засобів інформаційної безпеки, є технічними варіаціями вендорів і відрізняються за структурою, здатностям масштабування, функціональним властивостям і кількості розв'язуваних завдань.

Оцінити переваги SIEM-рішення допоможе аналіз по основних характеристиках.

Джерела й обробка подій

Чим більше джерел подій підтримує система, тим ефективніше захист. При цьому важливо, щоб SIEM-система забезпечувала індивідуальний підхід до нормалізації кожної події з різних джерел.

Роботу із програмою полегшує розбивка подій по категоріях. Синтаксичний аналіз інформаційних потоків (парсинг) подібних рішень реалізується за допомогою позначення найбільш критичних полів. Обновляються парсери, як правило, одночасно із впровадженням доповнень або змін системи.

Автознаходження, а також періодичне відновлення джерел експерти відносять до переваг. Однак єдиної думки з питання відновлення SIEM-рішення не існує. Відсутність автооновлення аналізаторів вендори іноді пояснюють захистом від змін логіки аналізу й пропонують проводити зміни SIEM під контролем власних фахівців. Такий підхід збільшує вартість володіння системою.

Разом: варто вибирати рішення, що взаємодіє з максимальною кількістю різнорідних систем, які використовуються в компанії. Багаторівнева платформа обробки інцидентів прискорить роботу із джерелами й легко адаптується до програмного забезпечення. Невисокі вимоги до апаратно-програмних засобів при

цьому будуть додатковою перевагою. Вітчизняні розроблювачі реалізують в SIEM підтримку джерел подій українського походження, яких немає в іноземних конкурентів, що також стане плюсом для замовників з України.

Збір інцидентів

Ефективна SIEM – це платформа з функціями нормалізації, об'єднання й фільтрації інцидентів. Перевагою буде обробка й зберігання raw-подій. Швидкість процесів при цьому на загальну картину не впливає. Маскування відомостей, моніторинг мережевого трафіку – функції допоміжні, але не марні.

Перевірити коректність роботи нормалізації, фільтрації й агрегації можливо на етапі тестування SIEM в «бойовому» режимі. Тому більше довіри викликають виробники, які надають безкоштовний тест-драйв повнофункціональної версії продукту.

Кореляція

Оптимальне SIEM-рішення зіставляє події в режимі реального часу, уміє проводити поведінковий аналіз і порівняння історичних даних.

Гнучкі налаштовування системи кореляції, збагачення інцидентів у коннекторі або в консолі керування, додаткова функція у вигляді ручної перевірки, можливість одночасної роботи з усіма механізмами – відмінні риси вдалої SIEM.

Візуалізація

Звітність SIEM-систем найчастіше формується у вигляді графіків, гістограмм і таблиць. Більшість звітів експортуються у файли п'яти форматів: MS Excel, RTF, PDF, CSV, HTML.

Комфортну роботу ІБ-фахівцю забезпечить русифікований інтерфейс. Це не принциповий критерій вибору, але за інших рівних умов – вигідна відмінність.

Загальні налаштовування й убудований функціонал

Зручність роботи з SIEM залежить насамперед від наявності убудованих умов кореляції подій, графічних панелей і шаблонів звітів. Чим більше убудованих кореляційних ресурсів, тим менше кваліфікованої, а значить –

платної допомоги від сторонніх фахівців буде потрібно при обслуговуванні платформи.

При розробці даного програмного продукту були проаналізовані типові завдання клієнтів з різних галузей в України. Це допомогло розробити набір передвстановлених політик, щоб замовники одержали перші аналітичні результати «з коробки» – відразу після установки, а не через 5-6 місяців після впровадження й тонкого налаштовування. При додаванні нових джерел подій система одержує оновлений набір передвстановлених правил.

Зручність застосування

Важливий маркер зручності роботи з SIEM – можливість централізовано координувати компоненти платформи з єдиної консолі, а також автоматично обновляти передвстановлені політики й шаблони звітності. Все це полегшить працю фахівця з інформаційної безпеки.

Ще один плюс на користь рішення – оперативність і якість технічної підтримки. По цьому параметрі в більшості випадків виграють вітчизняні виробники, головним чином, через невисоку вартість.

Разом

Оцінка SIEM по основних характеристиках забезпечує вибір рішення на попередньому етапі. Параметри «успішності» у різних замовників збігаються почасти. Більше глибоке порівняння SIEM-систем ураховує потреби й особливості IT-інфраструктури конкретної компанії, важливість параметрів і значимість функцій системи оцінюється індивідуально. Замовник самостійно формулює перелік критеріїв «успішного» рішення під час тестування. Специфічні параметри SIEM-системи будуть остаточно встановлені тільки в процесі повсякденної експлуатації.

Навіщо впроваджувати SIEM?

– За допомогою SIEM ваша організація зможе підвищити рівень захищеності інформаційних систем від зовнішніх і внутрішніх погроз ІБ.

– За рахунок запобігання й/або оперативного реагування на інциденти ІБ ваша організація зможе знизити й повністю виключити збиток від кібер-атак.

– З'явиться можливість одержання даних для процесів аналізу й керування ризиками ІБ.

– Для зниження операційних витрат за рахунок автоматизації процесів обробки й керування подіями ІБ від різних джерел.

– Для виконання ретроспективного аналізу інцидентів ІБ.

– Для проведення аналізу ефективності вжитих заходів по ІБ.

– Для своєчасного виявлення несанкціонованих змін в інформаційних системах.

– Щоб виявити слабкі точки захисту.

– Для своєчасного виявлення факти корпоративного шахрайства.

– Для можливості формування доказової бази при розслідуванні інцидентів.

– Для своєчасного виявлення й реагування на збої в роботі ІТ і ІБ-систем.

3.4 Розробка діаграми процесів

Розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3. Основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як співвідношення різних частин кожної з сукупностей. Склад статистичної сукупності графічно може бути представлений як за допомогою абсолютних, так і відносних показників. Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування).

Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграми потоків даних містять чотири типи елементів:

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

Рисунок 3.3 – Діаграма взаємодії процесів

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо реалізацію магістерської дипломної роботи. Були проведені розрахунки і підібрані набори тестових даних для перевірки правильності реалізації проектних рішень.

Було створено блок-схеми роботи системи. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується.

Блок-схеми показують весь процес роботи системи з підсистемами та частково доказують правильність вибраних проектних рішень. Тому від точності і детальної блок-схеми залежить результат всієї програми. При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає високого рівня декомпозиції задач на класи. На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підсистеми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки

програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

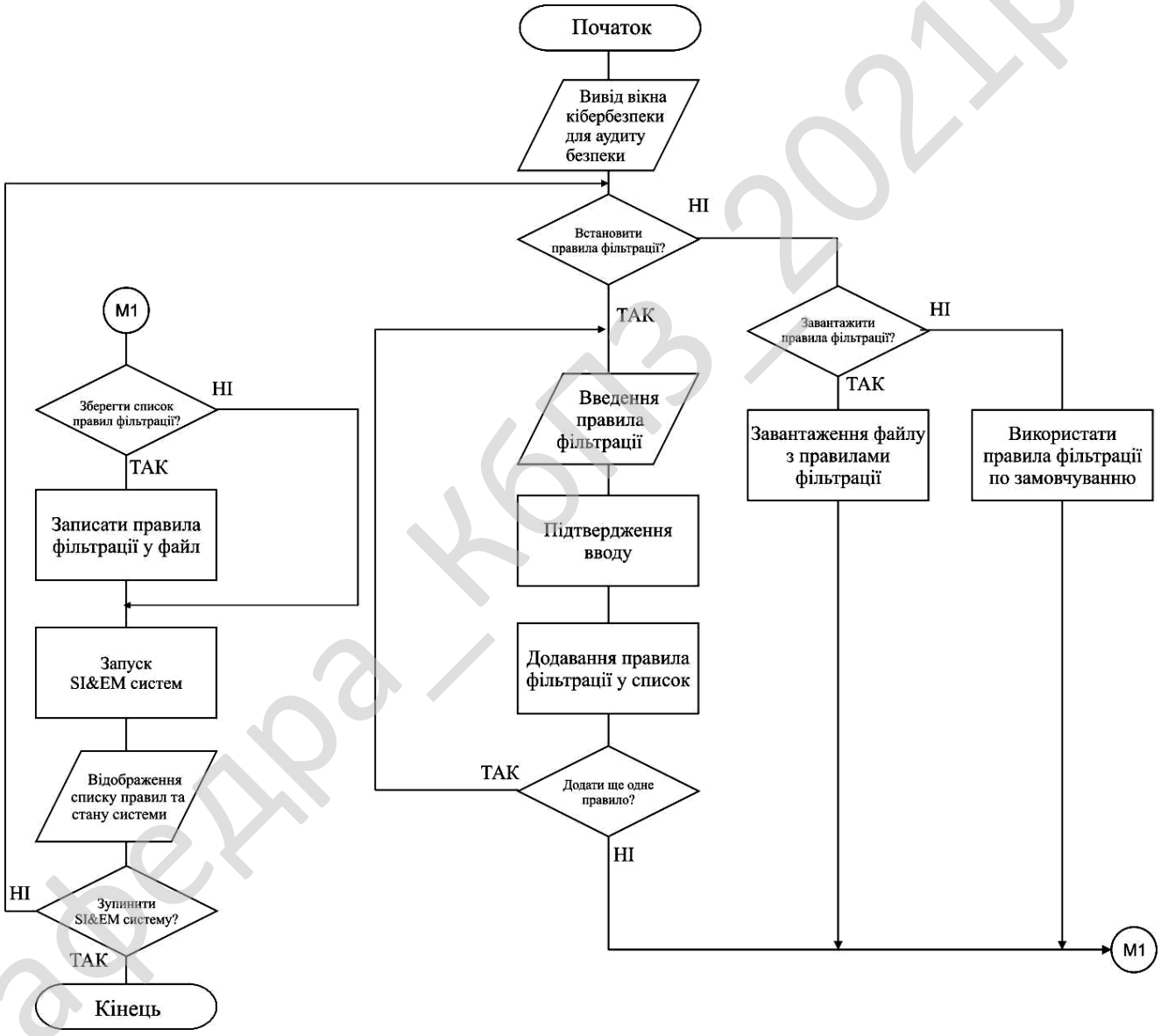

Рисунок 4.1 – Блок-схема основної програми

Було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу).

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Дія є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів.

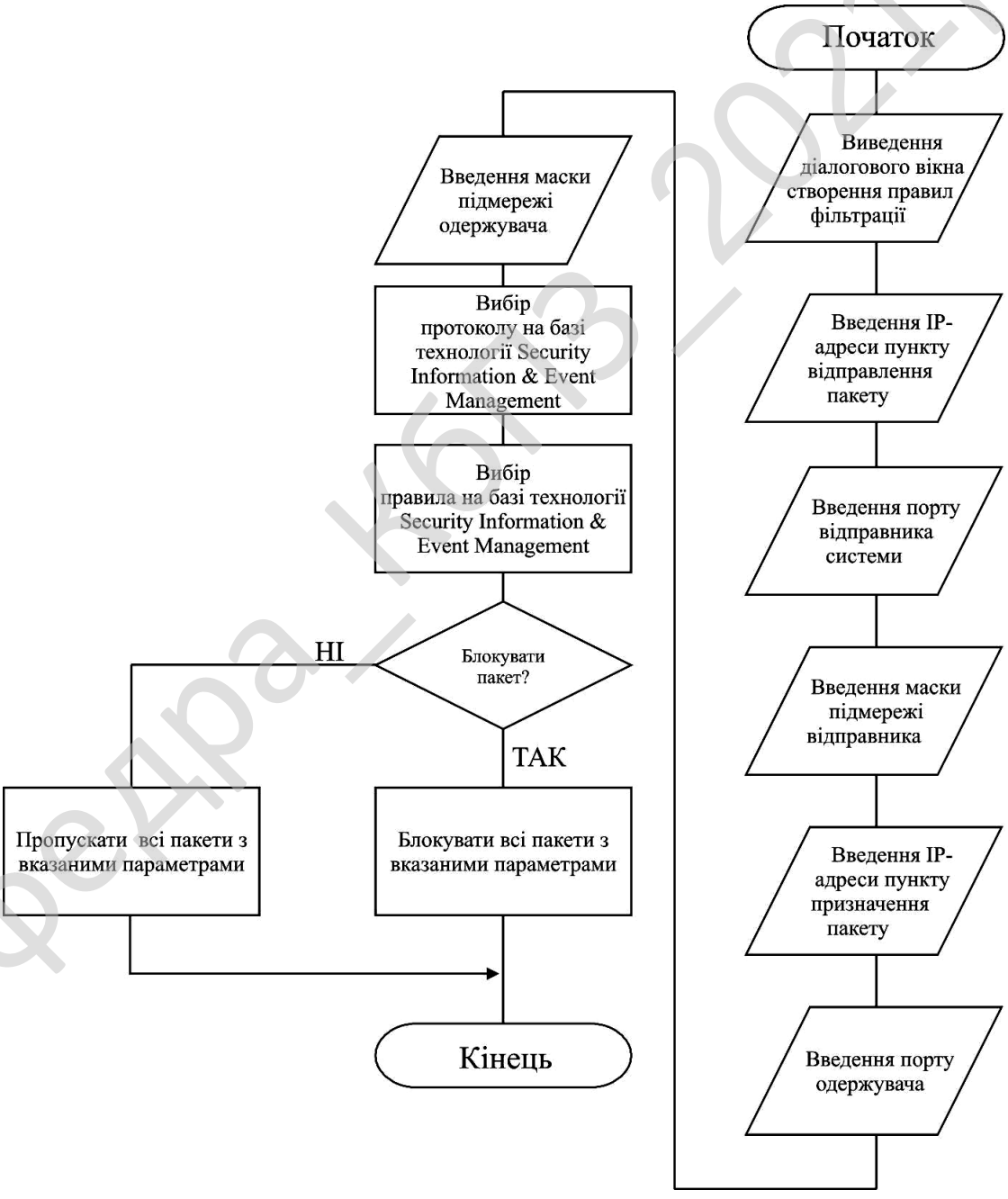

Рисунок 4.2 – Блок-схема роботи підпрограми

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграма прецедентів це діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором.

При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме – за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML – це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного).

Відношення розширення (extend) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

Jira – була використана комерційна система відслідковування помилок, призначена для організації взаємодії з користувачами, хоча в деяких випадках використовується і для управління проектами. Розроблено компанією Atlassian, є одним з двох її основних продуктів (поряд з вікі -системою Confluence). Має веб-інтерфейс.

Назва системи отримано шляхом усічення слова «Gojira» – Японського імені монстра Годзилла, що, в свою чергу, є відсиланням до назви конкуруючого продукту – Bugzilla; створювалася в якості заміни Bugzilla і багато в чому повторює її архітектуру. Система дозволяє працювати з декількома проектами. Для кожного з проектів створює і веде схеми безпеки і схеми оповіщення.

До версії 3.13.5 (включно) розрізнялися редакції Enterprise, Professional і Standard, після – Залишилася тільки редакція Enterprise (для великих організацій).

Система заснована на Java EE і працює на кількох популярних системах управління базами даних і операційних системах.

Основний елемент обліку в системі – завдання (ticket або issue). Завдання містить назву проекту, тему, тип, пріоритет, компоненти і зміст. Завдання може бути розширена додатковими полями (також і нові призначені для користувача поля можуть бути визначені), додатками (наприклад – Фотографіями, скриншотами) або коментарями. Завдання може редагуватися або просто змінювати статус, наприклад, з «відкритий» в «закритий». Які переходи між станами можливі, визначається через настроюється потік операцій. Будь-які зміни в задачі записуються в журнал.

Jira має велику кількість можливостей конфігурації: для кожної програми може бути визначений окремий тип завдання з власним workflow, набором статусів, одним або декількома видами уявлення (screens). Крім того, за допомогою так званих «схем» можна визначити для кожного індивідуального Jira-проекту власні права доступу, поведінку і видимість полів і багато іншого.

Завдяки універсальному підходу можна пристосувати Jira для багатьох непрофільних завдань, наприклад, керування вимогами, керування ризиками, аж до реалізації невеликої системи бронювання, автоматизації процесу рекрутингу.

Для інтеграції з зовнішніми системами підтримує інтерфейси SOAP, XML-RPC і REST. Поставляється із засобами інтеграції з такими системами управління версіями, як Subversion, CVS, Git, Clearcase, Team Foundation Server, Mercurial і Perforce.

Існують доповнення, що дозволяють вбудувати Jira в інтегровані середовища розробки, в тому числі Eclipse і IntelliJ IDEA. Перекладена багатьма мовами, включаючи російську, англійську, японську, німецьку, французьку, іспанську.

Для сторонніх розробників надаються кошти розробки розширень системи – плагінів. Розробники розширень можуть викладати плагіни для продажу на спеціальний розділ сайту Atlassian.

Є комерційним продуктом, який може бути ліцензований для роботи на локальному сервері або доступний в якості віддаленого додатки. Ціноутворення залежить від максимального числа користувачів, при цьому близько $50 за користувача для локального і $7 на місяць за користувача для віддаленого доступу є типовими цінами.

Для академічних і комерційних клієнтів доступний повний вихідний код під ліцензією розробника.

Для проектів з відкритим вихідним кодом Atlassian надає спеціальну безкоштовну ліцензію при дотриманні наступних правил:

– проект використовує ліцензії, схвалені Open Source Initiative;

– Вихідний код проекту доступний для скачування;

– у проекту є публічно доступна веб-сайт;

– програмне забезпечення від Atlassian є на веб-сайті проекту.

При розробці ПЗ було використано V-Model (або VEE модель) є моделлю розробки інформаційних систем (ІС), спрямованої на спрощення розуміння складнощів, пов'язаних з розробкою систем. Вона використовується для визначення єдиної процедури розробки програмного забезпечення, апаратного забезпечення та людино-машинного інтерфейсу.

Концепція V-подібної моделі була розроблена Німеччиною та США в кінці 1980-х років незалежно один від одного:

– Німецька V-модель була розроблена аерокосмічної компанією IABG в Оттобрунні поряд з Мюнхеном у сприянні з Федеральним департаментом з закупівлі озброєнь в Кобленці, для Міністерства оборони Німеччини. Модель була прийнята німецькою федеральною адміністрацією для цивільних потреб влітку 1992.

– Американська V-Model (VEE) була розроблена національною радою з системної інженерії (міжнародна – з 1995 року) для супутникових систем, включаючи обладнання, програмне забезпечення та взаємодію з користувачами.

Сучасною версією V-Model є V-Model XT, яка була затверджена в лютому 2005 року. V-модель використовується для управління процесом розробки програмного забезпечення для німецької федеральної адміністрації.

Зараз вона є стандартом для німецьких урядових і оборонних проектів, а також для виробників ПЗ в Німеччині. V-Model являє собою скоріше набір стандартів у галузі проектів, що стосуються розробки нових продуктів. Ця модель багато в чому схожа з Prince2 і описує методи як для проектного управління, так і для системного розвитку.

Основні принципи

Основний принцип V-подібної моделі полягає в тому, що деталізація проекту зростає при русі зліва направо, одночасно з плином часу, і ні те, ні інше не може повернути назад. Ітерації в проекті виробляються по горизонталі, між лівою і правою сторонами літери.

Стосовно до розробки інформаційних систем V-Model – варіація каскадної моделі, в якій завдання розробки йдуть зверху вниз по лівій стороні букви V, а завдання тестування – вгору по правій стороні букви V. Усередині V проводяться горизонтальні лінії, що показують, як результати кожної з фаз розробки впливають на розвиток системи тестування на кожній із фаз тестування.

Модель базується на тому, що приймально-здавальні випробування ґрунтуються, насамперед, на вимогах, системне тестування – на вимогах та архітектури, комплексне тестування – на вимогах, архітектурі та інтерфейсах, а компонентне тестування – на вимогах, архітектурі, інтерфейсах та алгоритмах

Цілі

V-модель забезпечує підтримку у плануванні та реалізації проекту. В ході проекту ставляться такі завдання:

1. Мінімізація ризиків: V-подібна модель робить проект більш прозорим і підвищує якість контролю проекту шляхом стандартизації проміжних цілей і опису відповідних їм результатів та відповідальних осіб. Це дозволяє виявляти

відхилення в проекті і ризики на ранніх стадіях і покращує якість управління проектом.

2. Підвищення та гарантії якості: V-Model – стандартизована модель розробки, що дозволяє домогтися від проекту результатів бажаної якості. Проміжні результати можуть бути перевірені на ранніх стадіях. Універсальне документування полегшує читаність, зрозумілість та контрольованість.

3. Зменшення загальної вартості проекту: Ресурси на розробку, виробництво, управління і підтримку можуть бути заздалегідь прораховані та проконтрольовані. Отримувані результати також універсальні і легко прогнозуються. Це зменшує витрати на подальші стадії та проекти.

4. Підвищення якості комунікації між учасниками проекту: Універсальний опис усіх елементів та умов полегшує взаєморозуміння всіх учасників проекту. Таким чином, зменшуються неточності у розумінні між користувачем, покупцем, постачальником і розробником.

Переваги:

– Користувачі V-Model беруть участь у розробці та підтримці V-моделі. Комітет з контролю за змінами підтримує проект і збирається раз на рік для обробки всіх отриманих запитів на внесення змін до V-Model.

– На старті будь-якого проекту V-подібна модель може бути адаптована під цей проект, так як ця модель не залежить від типів організацій та проектів.

– V-model дозволяє розбити діяльність на окремі кроки, кожен з яких буде включати в себе необхідні для нього дії, інструкції до них, рекомендації та докладне пояснення діяльності.

Обмеження. Наступні моменти не враховуються в V-моделі, але можуть бути розглянуті окремо, або можливо адаптувати модель під них:

– Не регулюється розміщення контрактів на обслуговування.

– Організація і виконання управління, обслуговування, ремонту та утилізації системи не враховуються в V-моделі. Однак, планування і підготовка до цих операцій моделлю розглядаються.

– V-подібна модель більше стосується розробки програмного забезпечення в проекті, ніж всієї організації процесу.

Переваги:

– У моделі особливе значення надається плануванню, спрямованому на верифікацію та атестацію розроблювального продукту на ранніх стадіях його розробки. Фаза модульного тестування підтверджує правильність деталізованого проектування. Фази інтеграції та тестування реалізують архітектурне проектування або проектування на вищому рівні. Фаза тестування системи підтверджує правильність виконання етапу вимог до продукту і його специфікації.

– У моделі передбачені атестація та верифікація всіх зовнішніх і внутрішніх отриманих даних, а не тільки самого програмного продукту.

– У V-подібної моделі визначення вимог виконується перед розробкою проекту системи, а проектування ПЗ – перед розробкою компонентів.

– Модель визначає продукти, які повинні бути отримані в результаті процесу розробки, причому кожні отримані дані повинні піддаватися тестуванню.

– Завдяки моделі менеджери проекту можуть відслідковувати хід процесу розробки, так як в даному випадку цілком можливо скористатися тимчасовою шкалою, а завершення кожної фази є контрольною точкою.

Недоліки:

– Модель не передбачає роботу з паралельними подіями.

– У моделі не передбачено внесення вимоги динамічних змін на різних етапах життєвого циклу.

– Тестування вимог в життєвому циклі відбувається занадто пізно, внаслідок чого неможливо внести змін, не вплинувши при цьому на графік виконання проекту.

– У модель не входять дії, спрямовані на аналіз ризиків.

– Деякий результат можна отримати тільки при досягненні низу букви V.

Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних програм і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

– набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;

– набір клієнтів, які використовують сервіси, що надаються серверами;

– мережа, яка забезпечує взаємодію між клієнтами та серверами.

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів.

Дуже важливо ясно уявляти, хто або що розглядається як «клієнт». Можна говорити про клієнтський комп'ютер, з якого відбувається звернення до інших комп'ютерів. Можна говорити про клієнтське та серверне програмне забезпечення. Нарешті, можна говорити про людей, які бажають за допомогою відповідного програмного та апаратного забезпечення отримати доступ до тієї чи іншої інформації.

Загальноприйнятим є положення, що клієнти та сервери – це перш за все програмні модулі. Найчастіше вони знаходяться на різних комп'ютерах, але бувають ситуації, коли обидві програми – і клієнтська, і серверна, фізично розміщуються на одній машині; в такій ситуації сервер часто називається локальним.

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій:

– рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;

– прикладний рівень, який реалізує основну логіку ПЗ і на якому здійснюється необхідна обробка інформації;

– рівень управління даними, який забезпечує зберігання даних та доступ до них.

Дворівнева клієнт-серверна архітектура передбачає взаємодію двох програмних модулів – клієнтського та серверного. В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

– модель тонкого клієнта, в рамках якої вся логіка ПЗ та управління даними зосереджена на сервері. Клієнтська програма забезпечує тільки функції рівня представлення;

– модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта. Товстими клієнтами часто також називають пристрої з обмеженою потужністю: кишенькові комп'ютери, мобільні телефони та ін.

Типовим прикладом клієнт-серверної взаємодії є WWW. Існує величезна кількість веб-серверів, на яких розміщується та чи інша інформація. У найпростішому випадку ця інформація являє собою набір веб-сторінок, які можуть зберігатися на сервері у вигляді файлів, розмічених за допомогою мови розмітки HTML. Але ситуація, як правило, є складнішою; значна частина веб-ресурсів на сучасному етапі є динамічними, тобто вони не існують в заздалегідь підготовленому вигляді, а створюються безпосередньо в процесі обробки запиту від користувача.

Для того, щоб людина, яка працює в Інтернеті, могла переглянути ту чи іншу сторінку, на її комп'ютері повинно бути встановлено відповідне програмне забезпечення. Програми для перегляду веб-сторінок називаються браузерами.

Але, крім браузерів, до серверів можуть звертатися і інші клієнти, а саме – автономні програми. Вони можуть передбачати взаємодію з людиною, а можуть працювати в цілком автоматичному режимі. Типовим класом таких програм є роботи, призначені для автоматичного перегляду веб-ресурсів. Зокрема, роботи є важливим елементом пошукових систем і використовуються ними для перегляду сторінок і збору інформації про них.

Для запиту до веб-сервера клієнтська програма повинна задати місцезнаходження комп'ютера, на якому розміщується серверна програма, назву потрібного документа і, можливо, інші дані, які специфікують запит. Мережа забезпечує знаходження сервера і передачу йому клієнтського запиту. Серверні програми обробляють цей запит, відповідь пересилається по мережі клієнтові.

Трирівнева клієнт-серверна архітектура, яка почала розвиватися з середини 90-х років, передбачає відділення прикладного рівня від управління даними. Відокремлюється окремий програмний рівень, на якому зосереджується прикладна логіка ПЗ. Програми проміжного рівня можуть функціонувати під управлінням спеціальних серверів ПЗ, але запуск таких програм може здійснюватися і під управлінням звичайного веб-сервера. Нарешті, управління даними здійснюється сервером даних.

Для роботи з системою користувач використовує стандартне програмне забезпечення –звичайний браузер. Це позбавляє його необхідності завантажувати та інсталювати спеціальні програми (хоча інколи така необхідність все-таки виникає).

Але користувачеві слід надати в розпорядженні інтерфейс, який дозволяв би йому взаємодіяти з системою і формувати запити до неї. Форми, що визначають цей інтерфейс, розміщуються на веб-сторінках та завантажуються разом з ними.

Веб-оглядач формує запит та пересилає його до сервера, який здійснює обробку. При необхідності сервер викликає серверні програмні модулі, які забезпечують обробку запиту і в разі потреби звертаються до сервера даних.

Сервер даних здійснює операції з даними, що зберігаються в системі та складають її інформаційну основу. Зокрема, він може здійснити вибірку з інформаційної бази відповідно до запиту та передати її модулю проміжного рівня для подальшої обробки. Дані, з якими працює сервер даних, найчастіше організовані як реляційна база даних.

Найчастіше веб-сервер і серверні модулі проміжного рівня розміщуються на одному комп'ютері, хоч і являють собою окремі і логічно незалежні програмні модулі.

На сучасному етапі для програмування модулів проміжного рівня використовується мова серверних сценаріїв PHP, а для управління даними – СУБД MySQL. Таким чином, зв'язку PHP-MySQL слід розглядати як стандартний інструмент для створення порівняно простих інтерактивних веб-сайтів та систем електронної комерції; близько 90% комерційних систем сьогодні створюється саме на цій основі. Водночас як засоби управління даними, так і middleware-засоби можуть бути найрізноманітнішими. Так, для створення серверних програм, крім PHP, широко застосовуються Java, Perl, Python, Delphi.

Взагалі, технології створення розподілених, зокрема веб-програм, стрімко розвиваються. Слід згадати про технології EJB (Enterprise Java Beans), CORBA, а також про .NET – порівняно нову ініціативу компанії Microsoft. Для зберігання даних та їх передачі часто використовується так звана розширювана мова розмітки XML (Extensible Markup Language).

NetBIOS (Network Basic Input/Output System) – протокол для роботи в локальних мережах на персональних ЕОМ типу IBM/PC, розроблений у вигляді інтерфейсу, який не залежить від фірми–виробника. Був розроблений фірмою Sytek Corporation за замовленням IBM в 1983 році. Він включає в себе інтерфейс сеансового рівня (англ. NetBIOS interface), в якості транспортних протоколів використовує TCP і UDP.

Особливістю NetBIOS є можливість його роботи поверх різних протоколів, найпоширенішими/відомими з яких є NetBEUI, IPX і стек протоколів TCP/IP;

причому якщо старі версії Windows орієнтувалися на більш легкі в реалізації і менш ресурсомісткі NetBEUI і IPX, то сучасні Windows орієнтуються на TCP/IP.

При використанні NetBEUI і IPX NetBIOS сам забезпечує надійність доставки даних (функціональність SPX не використовувати), а при використанні TCP/IP надійність доставки забезпечує TCP, за що удостоївся окремого імені «NBT».

Інтерфейс NetBIOS являє собою типовий інтерфейс взаємодії програм (API) для забезпечення мережевих операцій введення–виведення і управління транспортним протоколом.

Програми, що використовують NetBIOS API інтерфейс, можуть працювати тільки при наявності протоколу, що допускає використання такого інтерфейсу.

NetBIOS також визначає протокол, що функціонує на сеансовому/транспортному рівнях моделі OSI. Цей протокол використовується протоколами нижчих рівнів, такими як NBFP (NetBEUI) і NetBT для виконання мережевих запитів вводу–виводу і операцій, описаних в стандартному интерфейсном наборі команд NetBIOS.

Тобто, NetBIOS сам не підтримує виконання файлових операцій. Ця функція покладається на протоколи нижчих рівнів, а сам NetBIOS забезпечує тільки зв'язок з цими протоколами і NetBIOS API–інтерфейс.

Пакет NETBIOS створений для використання групою ЕОМ. Підтримує як режим сесій (робота через з'єднання), так і режим дейтаграм (без встановлення з'єднання). 16–символьні імена об'єктів в NETBIOS розподіляються динамічно.

NETBIOS має власну DNS, яка може взаємодіяти з інтернетівський. Ім'я об'єкта при роботі з NETBIOS не може починатися з символу *.

Додатки можуть знайти через NETBIOS потрібні їм ресурси, встановити зв'язок і послати або отримати інформацію. NETBIOS використовує для служби імен порт 137, для служби дейтаграм – порт 138, а для сесій – порт 139. Будь–яка сесія починається з NETBIOS–запиту, завдання IP–адреси і визначення TCP–

порту віддаленого об'єкта, далі йде обмін NETBIOS–повідомленнями, після чого сесія закривається.

Сесія здійснює обмін інформацією між двома NETBIOS – додатками. Довжина повідомлення лежить в межах від 0 до 131 071 байт. Припустимо одночасне встановлення декількох сесій між двома об'єктами.

При організації IP–транспорту через NETBIOS IP–дейтаграмма вкладається в NETBIOS–пакет. Інформаційний обмін відбувається в цьому випадку без встановлення зв'язку між об'єктами. Імена NETBIOS повинні містити в собі IP–адреси. Так, частина NETBIOS–адреси може мати вигляд IP.XX.XX. XX. XX, де IP вказує на тип операції (IP через Netbios), а XX. XX. XX.XX – IP– адреса.

Система NETBIOS має власну систему команд (call, listen, hang up, send, receive, session status, reset, cancel, adapter status, unlink, remote program load) і примітивів для роботи з дейтаграммами (send datagram, send broadcast datagram, receive datagram , receive broadcast datagram).

Всі кінцеві вузли NETBIOS діляться на три типи:

– широкомовні («b») вузли;

– вузли точка–точка («p»);

– вузли змішаного типу («m»).

IP–адреса може асоціюватися з одним із зазначених типів. B–вузли встановлюють зв'язок зі своїм партнером за допомогою широкомовних запитів. P– і M–вузли для цієї мети використовують netbios сервер імен (NBNS) і сервер розподілу дейтаграм (NBDD).

NetBIOS забезпечує:

– реєстрацію і перевірку мережевих імен;

– встановлення і розрив з'єднань;

– зв'язок з підтвердженням доставки інформації;

– зв'язок без підтвердження доставки інформації;

– підтримку управління і моніторингу драйвера і мережевої карти.

4.2 Захист розробленого програмного забезпечення

Розроблене програмне забезпечення захистимо за допомогою алгоритму захисту інформації RSA. Спочатку необхідно обчислити пару ключів (секретний ключ і відкритий ключ). Для цього відправник електронних документів обчислює два більших простих числа P и Q, потім знаходить їхній добуток N = P*Q і значення функції φ(N) = (P-1)(Q-1). Далі відправник обчислює число E з умов E < φ (N), НЗД (E, φ(N)) = 1 і число D з умов D < N, E*D ≡ 1(mod φ (N).

Пари чисел (E, N) є відкритим ключем. Цю пару чисел автор передає партнерам по переписці для перевірки його цифрових підписів. Число D зберігається автором як секретний ключ для підписування.

Допустимо, що відправник хоче підписати повідомлення M перед його відправленням. Спочатку повідомлення M (блок інформації, файл, таблиця) стискають за допомогою геш-функції h(-) у ціле число m: m = h(M).

Потім обчислюють цифровий підпис S під електронним документом M, використовуючи геш-значення m і секретний ключ D: $S = m \pmod N$.

Пари (M, S) передається партнерові-одержувачеві як електронний документ M, підписаний цифровим підписом S, причому підпис S сформований власником секретного ключа D.

Після прийому пари (M, S) одержувач обчислює геш-значення повідомлення M двома різними способами. Насамперед, він відновлює геш-значення m', застосовуючи криптографічне перетворення підпису S з використанням відкритого ключа E: $m' = S^E \pmod N$.

Крім того, він знаходить результат гешування прийнятого повідомлення M с допомогою такої ж геш-функції h(-): m = h(M).

Якщо дотримується рівність обчислених значень, тобто $S^E(\text{mod}) = h(M)$, то одержувач визнає пару (M, S) справжньою.

Доведено, що тільки власник секретного ключа D може сформувати цифровий підпис S по документі M, а визначити секретне число D по відкритому

числу E не легше, ніж розкласти модуль N на множники.

Крім того, можна строго математично довести, що результат перевірки цифрового підпису S буде позитивним тільки в тому випадку, якщо при обчисленні S був використаний секретний ключ D, що відповідає відкритому ключу E. Тому відкритий ключ E іноді називають "ідентифікатором" того, хто підписав.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено розроблене у магістерської дипломної роботі система кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи: Функціональних кнопок ПЗ; Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші; Верхнього меню; Розділу обрання групи.

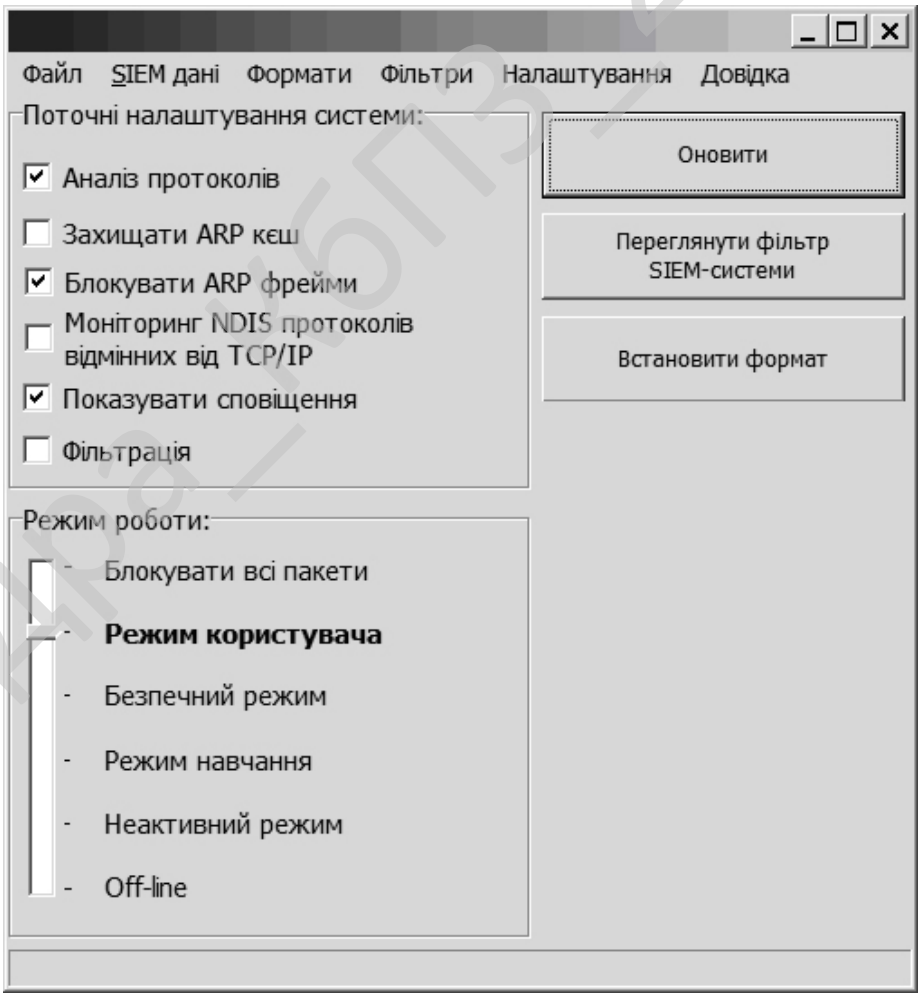

Рисунок 5.1 – Головне вікно ПЗ

Розроблена програма має дуже простий і зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий. Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

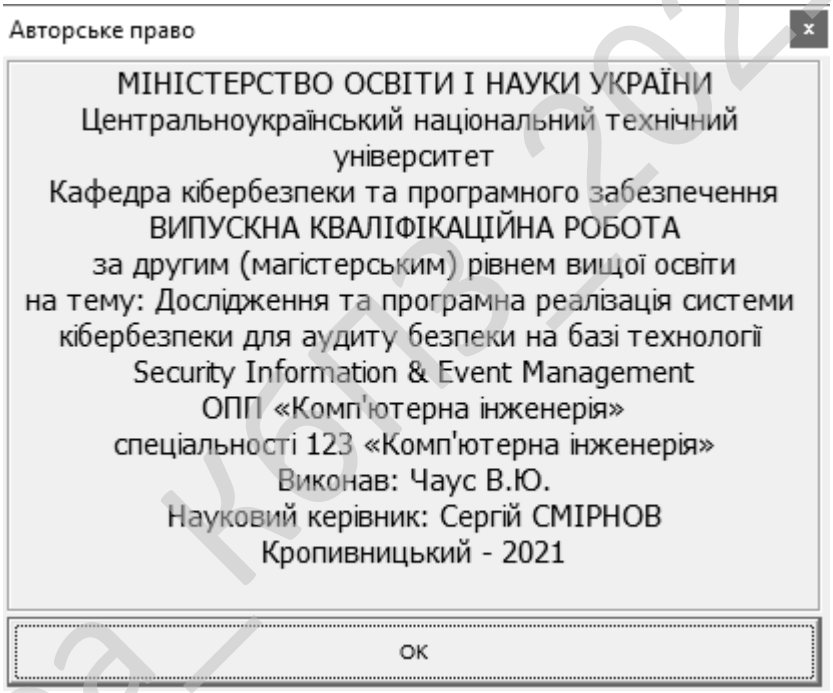

Рисунок 5.2 – Авторське право

SIEM здатна виявляти:

– мережеві атаки у внутрішньому й зовнішньому периметрах;

– вірусні епідемії або окремі вірусні зараження, невидалені віруси, бекдори й трояни;

– спроби несанкціонованого доступу до конфіденційної інформації;

– фрод і шахрайство;

– помилки й збої в роботі інформаційних систем;

– уразливості;

– помилки конфігурацій у засобах захисту й інформаційних систем.

Система SIEM універсальна за рахунок своєї логіки. Але для того щоб покладені на неї завдання вирішувалися – необхідні корисні джерела й правила кореляції. Будь-яка подія (наприклад, якщо в певній кімнаті відкрилися двері) можуть бути подані на вхід SIEM і використано.

Джерела вибираються на підставі наступних факторів:

– критичність системи (цінність, ризики) і інформації (оброблюваної й збереженої);

– вірогідність і інформативність джерела подій;

– покриття каналів передачі інформації (повинні враховуватися не тільки зовнішній, але й внутрішній периметр мережі);

– рішення спектру завдань ІТ і ІБ (забезпечення безперервності, розслідування інцидентів, дотримання політик, запобігання витоків інформації й т.п.).

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача.

Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик.

Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати IT рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене IT рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

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

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в IT рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути

невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

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

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

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

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

Проводилось тестування форматом білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

– Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Обрано умови розповсюдження – proprietary software. Програмне забезпечення, на яке зберігаються як немайнові, так і майнові авторські права. Отримавши або придбавши таке програмне забезпечення, користувач отримує обмежені права користування ним: може бути заборонено або закрито доступ до коду (вивчення), внесення змін, тиражування, розповсюдження та перепродаж. Програмне забезпечення вважається власницьким, якщо наявне хоча б одне з перелічених обмежень.

Найчастіше основним методом захисту майнових прав на власницьке ПЗ, поза ліцензійною угодою, власник обирає закриття сирцевого коду, захищаючи свій продукт від модифікації і вбудовуючи системи обмеження користування через авторизацію. Таке програмне забезпечення називається закритим. Проте, код власницького продукту може бути і відкритим, але власник може обмежити права користувача умовами користувацької ліцензії.

Власницьке програмне забезпечення та комерційне програмне забезпечення не є синонімами – власницьким може бути і безплатне (тобто, некомерційне) програмне забезпечення.

На противагу власницькому ПЗ існує вільне програмне забезпечення, автори і власники якого дозволяють вивчати, модифікувати і поширювати свій продукт.

Саме визначення власницького програмного забезпечення виникло в результаті діяльності громадського руху вільного програмного забезпечення (представленого Фондом вільного програмного забезпечення та іншими організаціями) і осмислення умов свободи користування програмами. Визначенням власницького програмного забезпечення є не невідповідність хоча б одній з базових умов вільного програмного забезпечення.

Сама назва власницьке ПЗ підкреслює визначальне значення власника у способі використання і можливостях розвитку цього програмного забезпечення.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

Метою розробки є дослідження та програмна реалізація системи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

Об'єктом дослідження є процес кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

Предметом дослідження є методи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

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

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

– Розроблено вітчизняний продукт кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

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

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці). В магістерській роботі було проведене дослідження та виконана програмна реалізація системи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management. Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

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

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	280 ((2 ост. цифри № зал +1)*10^1)
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

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

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	28000 ((2 ост. цифри № за +1)*10^3)
33. Норматив додаткової зарплати, % :	Нд	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

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

де: A – коефіцієнт Боема, $A = 2{,}45$; Size – загальний об'єм відлагодженого програмного коду, тис. рядків; B – показник ступеня, що визначається співвідношенням:

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

де: Wi – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}027.$$

$$T_{ном} = 2{,}45\cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

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

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

$$T_{уточн} = 6{,}78\cdot(0{,}88\cdot 0{,}93\cdot 0{,}88\cdot 0{,}91\cdot 0{,}95\cdot 1\cdot 1\cdot 0{,}87\cdot 1{,}22\cdot 1{,}16\cdot 1{,}1\cdot 1{,}1\cdot$$

$$\cdot 1{,}12\cdot 1{,}1\cdot 1\cdot 1{,}1\cdot 1{,}1) = 9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

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

де: C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4);

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%.

$$T_{РП} = 0{,}3\cdot 3{,}23\cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)}\cdot 82 = 168 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	168	Ф 7.1-7.4
Впровадження	13	Д13
Всього	209	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

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

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

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

$$Ч = \frac{209 \cdot 1}{60 - 5} = 3,8 \text{ ставки.}$$

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

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

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

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	385	12	4620	77
Монітор	160	12	1920	32
Клавіатура	140	12	1680	28
Маніпулятор «мишка»	30	12	360	6
Принтер матричний	185	1	185	3
Принтер лазерний	355	2	710	12
Принтер струминний	300	1	300	5
Сканер	155	2	310	5
Концентратор-маршрутизатор	155	2	310	5
Кабельні господарства ЛОМ на 1 м. п.	2,5	100	250	4
Кабельне господарство електромережі	48	50	2400	40
Копіювальний апарат	285	2	570	10
Усього за рік:			$З_ч$	227

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$\Phi_{др}^{с} = \frac{З_ч \cdot n_{тіс}}{1{,}2}, \qquad (7.6)$$

$$Ф_{др}^{с} = \frac{227 \cdot 3}{1{,}2} = 567{,}5 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$$Ч_{ел} = 567{,}5/(60 \cdot 8) = 1{,}2 \text{ ставки.}$$

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

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

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

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Розміщення графіки і контенту на Інтернет сторінках	0,5	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

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

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	12000	36000
Продакт-менеджер	0,25	8000	6000
Інженер-програміст	3,8	8000	91200
Інженер-електронщик	1,2	6000	21600
Інженер-системотехнік	0,25	6000	4500
Адміністратор мережі	0,5	6000	9000
Системний програміст	0,25	6000	4500
Дизайнер WEB	0,25	8000	6000
Інженер-верстальник	0,25	6000	4500
Бухгалтер-економіст	0,5	10000	15000
Всього за період розробки	$R_{сп}$ = 8,25	-	$Ф_{роб}$ = 198300

Розрахуємо середньоденну зарплату одного виконавця:

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

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

$$З_{сд} = \frac{198300}{8,25 \cdot 60} = 401 \, грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

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

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

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць;

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

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

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» (м. Кіровоград) ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 800…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 25 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

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

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

$$I_{нв} = 8 \cdot 3500 = 28000 \text{ грн.}$$

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

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми Компбест за 06.11.21 – джерело https://compbest.com.ua/.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	Intel Core i5-4590 (4 ядра по 3.3-3.7GHz), MB cache	1750
Системна плата	MSI H81M-P33 6 x USB 2.0, 2 x USB 3.(VGA, DVI, 2 x PS/2, LAN (RJ-45), 5 x Audi(FireWire	1200
Відеокарта	nVidia GeForce GTX 660, 2 GB GDDR5, 192 bit	750
Жорсткий диск	240 GB SSD	1200
Оперативна пам’ять	Samsung DDR3-1333 8Gb PC3-10600R EC(Registered (M393B1K70CH0-CH9Q5)	900
DVD-привод	DVDRW Pioneer DVR-TD10RS SATA Slir Black Bulk (DVR-TD10RS)	416
Корпус	ATX Middle Tower FOXCONN Pro, 3GTLA 489, PSU 350W(FSP Brand: ATX-350PNR 12cm), black, (front bezel – black+light silve body material – 0.6mm), 80mm fan (rear 2xUSB2.0/AUDIO/MIC, Air Duct, Tool-les chassis design,Thermally Advantaged Chassis	911

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Кулер	–	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: Al Type Cards, black	220
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1 170/160, D-SUB, Wide)	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USE cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп'ютери	15	10947	16420,5	180625,5
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	-	-	-	0
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	199177

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

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відраху-вання, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	199177	-	-
Всього по групі	199177	50	99588,5
Група 5, 6			
4. Вимірювальні пристрої	5190	25	1297,5
5. Транспортні засоби	0	20	0,0
6. Господарський інвентар	28000	25	7000
Всього по групі	33190	-	8297,5
7. Нематеріальні активи	120000	10	12000
Разом	$К_р$ = 1760367		$А_р$ = 190286

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

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

де: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 401 \cdot 209/280 = 300 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

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

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

$$З_д = 300 \cdot 10 \cdot 0{,}01 = 30 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 22\%$ від суми основної та додаткової зарплати:

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

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

$$С_{оц} = 0{,}01 \cdot 22(300+30) = 73 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $Н_г = 15\%$ від основної зарплати:

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

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 300 \cdot 15 \cdot 0{,}01 = 45 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.; $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.; $З_{М3}$ – вартість фарби, картріджей, тонеру, грн.; $N_е$ – кількість екземплярів програм, шт.

Згідно виданих викладачем норм приймаємо одну пачку паперу на три місяці розробки ($n_р$=0,33). Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$ = 105 грн., визначаємо вартість паперу за період розробки $N_м$ = 3 міс:

$$З_{М1} = Ц_п \cdot n_р \cdot N_м. \quad (7.16)$$

$$З_{М1} = 105 \cdot 0{,}33 \cdot 3 = 105 \text{ грн.}$$

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

$$З_{М2} = \sum Ц_д\ , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 2 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 2 грн./шт.

$$З_{М2} = 280 \cdot 12 = 3360 \text{ грн.}$$

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

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574+558+570 = 1702 \text{ грн.}$$

$$З_М = (105 + 3360 + 1702)/280 = 18 \text{ грн.}$$

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

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

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

$$О_п = 300 \cdot 15 \cdot 0{,}01 = 45 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 280 прим.):

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

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

$$A_м = 190286 \cdot 3/(280 \cdot 12) = 170 \text{ грн.}$$

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

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

$$C_п = 300+30+73+45+18+45+170 = 681 \text{ грн.}$$

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

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1. Основна зарплата виконавців	$З_о$	300
2. Додаткова зарплата виконавців	$З_д$	30
3. Відрахування на соціальні потреби	$C_{оц}$	73
4. Загальногосподарські витрати	$Г_{осп}$	45
5. Витрати на матеріали	$З_М$	18
6. Освоєння нових операційних систем, мов програмування	$О_п$	45
7. Амортизація основних фондів	$A_м$	170
8. Повна собівартість програмного забезпечення	$C_п$	681
9. Плановий прибуток	$П_р$	341
10.Ціна підприємства $Ц_п = C_п + П_р$	$Ц_п$	1022
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	204,4
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	1226,4

Визначимо плановий прибуток за рівнем рентабельності (P_n) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 50%.

$$П_р = 0{,}01 \cdot P_n \cdot C_n, \quad (7.22)$$

де: P_n – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 681 = 341 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

Об'єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

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

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	1226
Всього капітальних витрат	–	1226

7.7 Визначення експлуатаційних витрат

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

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування системи	$З_р$	15190	3032
2. Витрати на електроенергію	$З_{ел}$	1488	1030
3. Витрати на амортизацію	$З_{ам}$	0	307
Всього витрат за рік	$І$	16678	4369

Витрати на профілактичні роботи:

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

де: $Т_р$ – кількість годин обслуговування кожного комп'ютера за рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення витрати на обслуговування системи зменшились з 15190 грн до 3032 грн на рік.

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

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	1226	–	306,5
Всього відрахувань	-	–	1226	–	306,5

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

$$З_{ел} = П_{ел} \cdot T_р \cdot Ц_{ел}. \quad (7.24)$$

$$З_{ел\ баз} = 0{,}545 \cdot 1300 \cdot 2{,}1 = 1487{,}85 \text{ грн.}$$

$$З_{ел\ нов} = 0{,}545 \cdot 900 \cdot 2{,}1 = 1030{,}05 \text{ грн.}$$

7.8 Визначення економічної ефективності програмної продукції

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

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

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{р_m} \cdot K_{р_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (1022 - 681) \cdot 280 - (0{,}05 \cdot 1408000 + 0{,}5 \cdot 199177 +$$
$$+0{,}25 \cdot 33190 + 0{,}1 \cdot 28000) \cdot 3/12 = 50208 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

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

де: $K_р$ – балансова вартість основних фондів розробника.

$$T_в = \frac{1760367}{(1022 - 681) \cdot 280 \cdot 12 / 3} = 4{,}6 \text{ роки}$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

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

$$E_{сп} = (16678\text{-}4369)\text{-}0{,}25 \cdot 1226 = 12003 \text{ грн.}$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	280
2. Повна собівартість розробленої програми	Грн.	681
3. Ціна розробленої програми	Грн.	1022
4. Плановий прибуток від реалізації розробленої програми	Грн.	341
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1760367
7. Загальний прибуток від реалізації програмної продукції	Грн.	95480
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	50208
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Роки	4,6
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	1226
11.Величина економічного ефекту у користувача програмної продукції	Грн.	12003
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,1

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \tag{7.28}$$

$$T_{сп} = \frac{1226}{16678 - 4369} = 0{,}1 \; року \, .$$

7.9 Висновки

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

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Протягом усієї історії людство приділяє прискіпливу увагу безпеці життя. Охорона праці є складовою частиною безпеки життя.

Законом України “Про охорону праці” [1] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».

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

Ці шкідливі фактори можуть привести до професійних захворювань [2].

При розгляді шкідливих чинників роботи програмістів та інших спеціалістів ІТ будемо керуватись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98, та

«Правила охорони праці під час експлуатації електронно-обчислювальних машин» НПАОП 0.00-1.28-10,

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення впливу комп'ютера на організм програміста визначимо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста,

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Програміст працює з електронно-обчислювальною машиною (ЕОМ) та іншим обладнанням, яке є джерелом небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. Так як програміст постійно перебуває в приміщенні, тому для комфортних умов праці в цьому приміщенні необхідно створити належний мікроклімат.

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території.

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

– монотонність праці;

– електромагнітні (у т.ч. високочастотні) випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шуми;

– статичні навантаження на кістково-м’язовий апарат;

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

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

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	5
Довжина	7
Висота	2,9

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	7
Обсяг, V	$м^3$	не менше 20.0	20,3

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин).

У зазначеному приміщенні працює 5 осіб. За даними, які наведено у табл. 8.1 та табл. 8.2, можна зробити висновок, що площа та об'єм приміщення у розрахунку на одно робоче місце програміста відповідають нормативним вимогам (Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»).

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, робота, яка виконується в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря у приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У

приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура, °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22,5-23,5	40-55	0,12
Тепла	23-25	50-70	0,1	23-24	52-70	0,11

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

В літню пору застосовується кондиціонер.

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

У приміщенні знаходяться наступні джерела шуму: принтер *HP Laser 135a*, електродвигуни вентиляторів ЕОМ.

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

Працю працівника, який постійно працює за комп’ютером, згідно ДБН В.2.5 – 28 – 2006 р. можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V -го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному

освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 лк. Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значень;

Таким чином можна припустити, що основною причиною можливого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов’язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга).

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, бажано мати під рукою дефібрилятор і підготовлений персонал для роботи з ним.

8.5 Розрахункова частина

Початкові данні для розрахунку захисного заземлення:

Тип заземлення: робоче заземлення нульової точки трансформатора. Заземленню підлягає виробниче обладнання організації. Напруга – 220/380 В. Розташування заземлюючих електродів – по контуру.

Розрахунок проводиться за допустимим опором розтіканню струму заземлювача методом коефіцієнта використання заземлювачів.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта – чернозем. Товщина верхнього шару грунта: H=1 м. (див. Рис.8.1). Тип нижнього шару грунта – суглинок полутвердий. Для захисного заземлення: застосовуються ветикальні електроди – пруток довжиною L = 2 м. Відстань між вертикальними заземлювачами (електродами) A=2 м. Діаметр ветикальньного електрода (прутка) D=20 мм, Тип горизонтального заземлювача: металева полоса. Розміри перетину з'єднуючої полоси: 80 х 6 мм. (b=80 мм.). Глибина закладення контура заземлення t= 0,7 м. (див. Рис.8.1). Опір заземлювача, який нормується: $R_{ЗН}$ = 4 Ом.

Розрахунок захисного заземлення можна автоматизувати за допомогою програми, сирцевий код якої опублікован на стр.13-16 [3].

Розрахунок.

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,7+2/2=1,7 \text{ м.}$$

Еквівалентний питомий опір грунта:

$$\rho_{екв}= \psi \rho_1 \rho_2 L /[\rho_1 \psi(L-H+t)+ \rho_2 \psi(H-t)]=109,59 \text{ Ом}$$

де

ψ = 1,4 табличне значення коефіцієнта сезонності у багатошаровому грунті;

ρ_1= 200 Ом*м. – табличне значення питомого опору верхнього шару грунта;

ρ_2= 30 Ом*м. – табличне значення питомого опору нижнього шару грунта.

Опір розтіканню електричного струму одного вертикального заземлювача:

$$R_O = [\rho_1/(2\pi*L)]*[(ln)2L/D)+0,5ln[(4T+L)/(4T-L)]=45,77 \text{ Ом.}$$

Опір розтіканню електричного струму заземлювача, який нормується, якщо $\rho_{екв}$ >100 (у нас $\rho_{екв}$ =109,59 >100):

$$R= R_{ЗН} *\rho_{екв}/100=4*112,9/100=4,38 \text{ Ом.}$$

Опір розтіканню єлектричного струму горизонтального заземлювача (полоси):

$$R_П =0,366(\rho_{екв}\ \psi / L_П \cdot \eta_П)\lg(2/ L_П * L_П/bt)=39,17 \text{ Ом.}$$

де $\eta_П$ = 0,27 (табличне значення коефіцієнта використання горизонтального заземлювача, залежить від співвідношення A/L); довжина горизонтального заземлювача (полоси):

$L_П$=A*n =2*23=46 м. (при розташуванні заземлювачів по контуру);

де n – ітераційна кількість вертикальних заземлювачів.

Опір заземлювача розтіканню електричного струму :

$$R_B = R_П R / (R_П - R) =4,46 \text{ Ом.}$$

Кількість вертикальних заземлювачів:

$$n = R_O / R_B * \eta_C =23 \text{ шт.}$$

де η_C = 0,47 (табличне значення коефіцієнта використання вертикального заземлювача, залежить від співвідношення A/L).

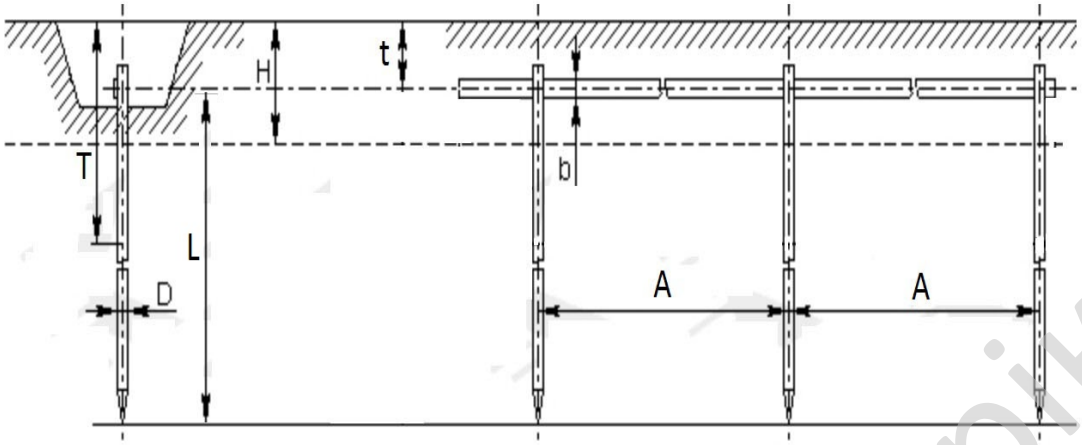

Рисунок 8.1 – Схема штучного заземлювача

8.6 Висновки до розділу

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

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров'я програміста. Розроблено заходи з охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

– Досліджена система кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

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

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

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

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows XP/Vista/7/8/10.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм RSA.

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

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 12003 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,1 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Чаус В.Ю. Дослідження та програмна реалізація системи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management // Збірник праць молодих науковців ЦНТУ. – Вип. 12. – Кропивницький: ЦНТУ, 2022.

2. Гмурман В.Е. Теория вероятностей и математическая статистика / В.Е. Гмурман. – М.: Высшая школа, 2003. – 479 с.

3. Смірнов О.А. Дисперсійний аналіз мережного трафіку для забезпечення інформаційної безпеки телекомунікаційних систем / О.О. Кузнецов, О.А. Смірнов, Д.О. Даниленко // Інформаційна та економічна безпека: сучасний стан та тенденції розвитку : монографія за заг. ред. – Х.: ХІБС УБС НБУ – 2014 – С. 82-100.

4. Смірнов О.А. Дослідження методів виявлення вторгнень в телекомунікаційні системи та мережі / Д.О. Даниленко, О.А. Смірнов, Є.В. Мелешко // Системи озброєння і військова техніка. – Випуск 1(29) – Х.: ХУПС – 2012. – С. 92-100

5. Смирнов А.А. Метод обнаружения вредоносного программного обеспечения. Часть 1. Корреляционный анализ сетевого трафика // А.А.Смирнов, Д.А. Даниленко, Е.В.Мелешко // Науково-технічний журнал «Інформаційно-керуючі системи на залізничному транспорті» – Випуск 4(95). – Х.: УкрДАЗТ – 2012. – С. 8-14.

6. Смирнов А.А. Методы обнаружения вредоносного программного обеспечения в телекоммуникационных системах и сетях / Д.А. Даниленко // Збірник наукових праць "Системи обробки інформації". – Випуск 3(101) том 2. – Х.: ХУПС – 2012. – С. 152-155.

7. Смирнов А.А. Системы обнаружения и предотвращения вторжений для защиты телекоммуникационных сетей от вредоносного программного

обеспечения / Д.А. Даниленко, А.А. Смирнов, А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 1 (21) том 2. – Київ: ДП «ЦНДІНУ». – 2012. – С. 183-186.

8. Смирнов А.А. Системы обнаружения и предотвращения вторжений для защиты компьютерных сетей от вредоносного программного обеспечения / Д.А. Даниленко, А.А. Смирнов, И.Г. Кирилов // Збірник тез доповідей науково-практичної конференції «Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку». м. Харків. 21-22 березня 2012 р. – Харків. АВВ МВС. – 2012. – С. 70-71.

9. Смірнов О.А. Дослідження методів виявлення вторгнень в телекомунікаційні мережі для підвищення інформаційної безпеки // Д.О. Даниленко // Збірник тез науково-практичної конференції «Захист інформації в інформаційно-комунікаційних системах». м. Київ. 24-27 квітня 2012 р. – Київ: НАУ. – 2012. – С. 22-25.

10. Смирнов А.А. Исследование систем обнаружения и предотвращения вторжений для защиты телекоммуникационных сетей от вредоносного программного обеспечения / Д.А. Даниленко // Збірник тез доповідей VIII наукової конференції «Новітні технології – для захисту повітряного простору». Харків. 18-19 квітня 2012 р. – м. Харків. ХУПС. – 2012. – С. 45.

11. Смирнов А.А. Исследование методов сигнатурного обнаружения вредоносного программного обеспечения в телекоммуникационных системах и сетях // Д.А. Даниленко // Збірник тез XIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 13-14 квітня 2012 р. – Кіровоград: КНТУ. – 2012. – С. 43-45.

12. Смирнов А.А. Исследование методов проактивной защиты от вредоносного программного обеспечения в телекоммуникационных системах и сетях / Д.А. Даниленко // Збірник тез V міжнародної науково-практичної конференції «інтегровані інтелектуальні робототехнічні комплекси» (ІПРТК-2012). м. Київ. 15-16 травня 2012 р. – Київ: НАУ. – 2012. – С. 314-315.

13. Смирнов А.А. Метод обнаружения вредоносного программного обеспечения на основе корреляционного анализа сетевого трафика / Д.А. Даниленко // Матеріали XII всеукраїнської наукової інтернет-конференції «Наукові дослідження: зв'язок теорії і практики». м. Тернопіль. 29-30 квітня 2012 р. – Тернопіль: ТНЕУ. – 2012. – С. 9-10.

14. Смирнов А.А. Метод детектирования вредоносного трафика в телекоммуникационных сетях на основе использования bds-тестирования / Д.А. Даниленко // Збірник тез V міжнародної науково-практичної конференції «Комп'ютерні системи та мережні технології» (CSNT-2012). м. Київ. 13-15 червня 2012 р. – Київ: НАУ. – 2012. – С. 121.

15. Смирнов А.А. Обнаружение и предотвращение вторжений в компьютерных сетях на основе статистического анализа сетевого трафика / А.А. Смирнов, Д.А. Даниленко // Збірник тез доповідей науково-практичної конференції «Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку». м. Харків. 12-13 березня 2014 р. – Харків. АВВ МВС. – 2014. – С. 13-14.

16. Смірнов О.А. дисперсійний аналіз мережного трафіку для забезпечення інформаційної безпеки телекомунікаційних систем та мереж / О.А. Смірнов, Д.О. Даниленко // Збірник тез V Всеукраїнської науково-практичної конференції "Інформатика та системні науки". м. Полтава. 13-15 березня 2014 р. – Полтава: ПУЕТ. – 2014. – С. 289-291.

17. Смирнов А.А. Метод дисперсионного анализа сетевого трафика для обнаружения и предотвращения вторжений в телекоммуникационных системах и сетях/ А.А. Смирнов, Д.А. Даниленко // Збірник тез VI міжнародної науково-практичної конференції "Проблеми і перспективи розвитку IT-індустрії". м. Харків. 17-18 квітня 2014 р. – Харків: ХНЕУ. – 2014. – С. 258.

18. Смірнов О.А. метод забезпечення інформаційної безпеки телекомунікаційних систем з використанням дисперсійного аналізу мережного трафіку / О.А. Смірнов, Д.О. Даниленко // Збірник тез міжнародної науково-

практичної конференції «Інформаційна та економічна безпека» (INFECO-2014)». м. Харків. 15-16 травня 2014 р. – Харків: ХІБС УБС НБУ. – 2014. – С. 135-139.

19. Девянин П.Н. Модели безопасности компьютерных систем / П.Н. Девянин. – М.:Издательский центр «Академия», 2005. – 144 с.

20. Домарев В.В. Безопасность информационных технологий. Методология создания систем защиты / В.В. Домарев. – К.: ООО "ТИД "ДС", 2002 – 688 с.

21. ДСТУ ISO/IEC TR 13243-2003 Інформаційні технології. Посібник із методів та механізмів якості послуг / [Электронный ресурс]. – Режим доступа к ресурсу: http://document.ua/informaciini-tehnologiyi.-posibnik-iz-metodiv-ta-mehanizmiv--nor2718.html

22. ДСТУ В 3265 – 95. Зв'язок військовий. Терміни та визначення. – К.: УкрНДІССІ, 1995. – 23 с.

23. ДСТУ ISO 9000:2007 Системи управління якістю. Основні положення та словник термінів [Электронный ресурс]. – Режим доступа к ресурсу: http://document.ua/docs/tdoc14237.php

24. Ершов В.А.. Мультисервисные телекоммуникационные сети / В.А. Ершов, Н.А. Кузнецов. – М.: МГТУ им. Н.Э. Баумана, 2003. – 432 с.

25. Закон України «Про захист інформації в інформаційно-телекомунікаційних системах» [Электронный ресурс]. – Режим доступа к ресурсу: http://zakon4.rada.gov.ua/laws/show/2594-15

26. Информационная война и защита информации. Словарь основных терминов и определений [Электронный ресурс]. – Режим доступа к ресурсу: http://www.csef.ru/files/csef/articles/2176/2176.pdf

27. Казарин О.В. Безопасность программного обеспечения компьютерных систем / О.В. Казарин. – М.:МГУЛ, 2003. – 212 с.

28. Касперский Е. Компьютерное зловредство / Е. Касперский. – СПб.: Питер, 2007. – 208 с.

і. Касперский К. Техника сетевых атак. [Електронний ресурс]. – Режим

доступу до ресурсу: http://rghost.ru/download/43730077/

29. 8e48b6263ce45c7dc2a65a7453383dc33b22486d/Крис%20Касперски%20-%20Техника%20сетевых%20атак.pdf

30. Касперский К. Техника и философия хакерских атак. / К. Касперский. – М.: Солон-Пресс, 2004 – 272 с.

31. Концепция (основы государственной политики) Национальной безопасности Украины, утвержденная постановлением Верховной Рады Украины от 16 января 1997 г. № 3/97-ВР. [Электронный ресурс]. – Режим доступа к ресурсу: http://search.ligazakon.ua/l_doc2.nsf/link1/ F970003.html

32. Конри-Мюррей Эндрю Выявление вторжений / «Журнал сетевых решений/LAN», № 01, 2001 [Электронный ресурс]. – Режим доступа к ресурсу: http://www.osp.ru/lan/2001/01/131532/

33. Кузнецов О.О. Протоколи захисту інформації у комп'ютерних системах та мережах / О.О. Кузнецов, С.Г. Семенов. – Х.: ХНУРЕ, 2009. – 184 с.

34. Кузнецов О.О. Методи обробки сигналів даних та зображень / О.О. Кузнецов, Г.А. Кучук, С.Г. Семенов. –Х.: НТУ «ХПІ», 2011. – 292 с.

35. Кузнецов А.А. Дисперсионный анализ сетевого трафика для обнаружения и предотвращения вторжений в телекоммуникационных системах и сетях / А.А. Кузнецов, А.А. Смирнов, Д.А. Даниленко // Збірник наукових праць "Системи обробки інформації". – Випуск 2(118). – Х.: ХУПС – 2014. – С. 81-85.

36. Лоскутов А.Ю. Основы теории сложных систем / А.Ю. Лоскутов, А.С. Михайлов. – М.-Ижевск: Институт компьютерных исследований, 2007. – 620 с.

37. Лукацкий А.В. Обнаружение атак / А.В. Лукацкий. – С.Пб.:BHV – Санкт – Петербург, 2003. – 596 с.

38. Методи підвищення оперативності передачі даних та захисту інформації у телекомунікаційній мережі: звіт про НДР (проміжний) / Наук. кер. О.А. Смірнов. – К.:КНТУ, 2013 № ДР 0112U006631

39. Методология функционального моделирования IDEF0. Руководящий

документ [Электронный ресурс]. – Режим доступа к ресурсу: http://www.scribd.com/doc/76782197/МЕТОДОЛОГИЯ-ФУНКЦИОНАЛЬНОГО-МОДЕЛИРОВАНИЯ-IDEF0

40. Моделирование технических систем с AllFusion Process Modeler (ранее BPwin) [Электронный ресурс]. – Режим доступа к ресурсу:: http://www.sdteam.com/t5506

41. Назаров А.Н. Модели и методы расчета структурно-сетевых параметров сети АТМ / А.Н. Назаров. – М.: Горячая линия – Телеком, 2002. –255 с.

42. Наиболее опасная страна. Попытка заражения ПК. [Электронный ресурс]. – Режим доступа к ресурсу:: http://sooweb.ru/news/naibolee_opasnaja_strana_popytka_zarazhenija_pk/ 2012-01-23-526

43. НД ТЗІ Вимоги до захисту інформації WEB-сторінки від несанкціонованого доступу [Электронный ресурс]. – Режим доступа к ресурсу: http://www.csk.kfc.in.ua/documents/nakaz-web.doc

44. НД ТЗІ 2.5 -004-99 Критерії оцінки захищеності інформації в комп'ютерних системах від несанкціонованого доступу. Затверджено наказом ДСТСЗІ СБ України від 28.04.1999р., №22. [Электронный ресурс]. – Режим доступа к ресурсу: http://do.gendocs.ru/docs/index-27508.html?page=8

45. Олифер В.Г. Компьютерные сети. Принципы, технологии, протоколы. 4-е изд. / В.Г. Олифер, Н.А. Олифер. – СПб.: Питер, 2012. – 943 с.

46. Поповский В.В. Защита информации в телекоммуникационных системах / В.В. Поповский, А.В. Персиков. – Х.: ООО "Компания СМИТ", 2006. Т2 – 292 с.

47. Д. Соломон, М. Руссинович. Внутреннее устройство Microsoft Windows 2000. Мастер-класс. / Пер. с англ. – СПб.: Питер; М.: Издательско-торговый дом «Русская Редакция», 2001.

48. В. Столлингс. Криптография и защита сетей: принципы и практика (2-е издание). / Пер. с англ. – М.: Издательский дом «Вильямс», 2001.

49. Баричев С. Г., Гончаров В. В., Серов Р. Е. Основы современной криптографии. – М.: Горячая линия – Телеком, 2001.

50. Операционные системы: Учебник для вузов. – 2-е изд. Гордеев Александр. Питер – 2006 – ISBN 5-94723-632-X.

51. Windows XP. Руководство пользователя. Берлинер Э., Глазырин Б., Глазырина И. Бином – 2004 – ISBN.5-9518-0092-7.

52. Основы технологии PKI. В. С. Горбатов, О. Ю. ПолянскаяИздательство: Горячая Линия – Телеком, 2004 г. – 248 стр. – ISBN 5-93517-154-6.

53. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

54. Про охорону праці: Закон України від 14.10.1992 р. № 2694-XII. Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12#Text

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

56. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. – Кіровоград: КІСМ, 1997. – 20 с. Ре жим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 42-13 від 02.08.2021 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

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

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows XP/Vista/7/8/10 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows XP/Vista/7/8/10.

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Visual C++.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Економічні вимоги

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

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2021 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинні бути розглянуті шкідливі і небезпечні фактори при роботі з комп’ютером.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 105 аркушів.

10 Етапи розробки

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

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

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2021 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 20.12.2021 р.

Додаток Б

(обов'язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

другим (магістерським) рівнем вищої освіти

_____ Смірнов С.А.

Дослідження та програмна реалізація системи кібербезпеки для аудиту безпеки на базі технології Security Information & Event Management

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 42

Літера: РП

Кропивницький – 2021 року

SIEM_AppDoc.cpp – формування правил

```
//Описувач класу програми
//Підключення основних оголошень діалогового вікна
#include "stdafx.h"
#include "SIEM_App.h"

#include "SIEM_AppDoc.h"

// Ініціалізація дебаг інформації
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CSIEM_AppDoc
//Мапінг Windows подій, перехоплення пост повідомлень
IMPLEMENT_DYNCREATE(CSIEM_AppDoc, CDocument)

BEGIN_MESSAGE_MAP(CSIEM_AppDoc, CDocument)
	//{{AFX_MSG_MAP(CSIEM_AppDoc)
		!
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CSIEM_AppDoc construction/destruction

//Опис конструктора
CSIEM_AppDoc::CSIEM_AppDoc()
{
	nRules = 0;
	defaultAction = PF_ACTION_FORWARD;
}
//Опис деструктора
CSIEM_AppDoc::~CSIEM_AppDoc()
{
}

//обробка подій (пост повідомлень) Windows
BOOL CSIEM_AppDoc::OnNewDocument()
{
	if (!CDocument::OnNewDocument())
		return FALSE;

	return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CSIEM_AppDoc serialization
//обробка подій (пост повідомлень) Windows
void CSIEM_AppDoc::Serialize(CArchive& ar)
{
	if (ar.IsStoring())
	{
	}
	else
	{
	}
}

/////////////////////////////////////////////////////////////////////////////
// CSIEM_AppDoc diagnostics
//опис функцій обробки дебаг інформації
#ifdef _DEBUG
```

```
void CSIEM_AppDoc::AssertValid() const
{
	CDocument::AssertValid();
}

void CSIEM_AppDoc::Dump(CDumpContext& dc) const
{
	CDocument::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CSIEM_AppDoc commands
//обробка подій (пост повідомлень) Windows по додаванню правила
int CSIEM_AppDoc::AddRule(unsigned long srcIp,
                                        unsigned long srcMask,
                                        unsigned short srcPort,
                                        unsigned long dstIp,
                                        unsigned long dstMask,
                                        unsigned short dstPort,
                                        unsigned int protocol,
                                        int action)
{

	if(nRules >= MAX_RULES)
	{
		return -1;
	}

	else
	{
		rules[nRules].sourceIp		= srcIp;
		rules[nRules].sourceMask	= srcMask;
		rules[nRules].sourcePort	= srcPort;
		rules[nRules].destinationIp	= dstIp;
		rules[nRules].destinationMask = dstMask;
		rules[nRules].destinationPort = dstPort;
		rules[nRules].protocol		= protocol;
		rules[nRules].action		= action;

		nRules++;
	}

	return 0;
}
//обробка подій (пост повідомлень) Windows по скиданню правил
void CSIEM_AppDoc::ResetRules()
{
	nRules = 0;
}
//обробка подій (пост повідомлень) Windows по видаленню правила
void CSIEM_AppDoc::DeleteRule(unsigned int position)
{
	// out of range
	if(position >= nRules)
		return;

		if(position != nRules - 1)
	{
		unsigned int i;

		for(i = position + 1;i<nRules;i++)
		{
			rules[i - 1].sourceIp		= rules[i].sourceIp;
			rules[i - 1].sourceMask		= rules[i].sourceMask;
			rules[i - 1].sourcePort		= rules[i].sourcePort;
			rules[i - 1].destinationIp	= rules[i].destinationIp;
			rules[i - 1].destinationMask	= rules[i].destinationMask;
			rules[i - 1].destinationPort	= rules[i].destinationPort;
```

```
                rules[i - 1].protocol         = rules[i].protocol;
                rules[i - 1].action                 = rules[i].action;
            }
        }
        nRules ---i;
    }
```

SIEM_App.cpp - головний файл програми

```
//Описувач головного класу програми
//Підключення основних оголошень діалогового вікна
#include "stdafx.h"
#include "SIEM_App.h"

#include "MainFrm.h"
#include "SIEM_AppDoc.h"
#include "SIEM_AppView.h"

// Ініціалізація дебаг інформації
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CSIEM_AppApp
//Мапінг Windows подій, перехоплення пост повідомлень
BEGIN_MESSAGE_MAP(CSIEM_AppApp, CWinApp)
	//{{AFX_MSG_MAP(CSIEM_AppApp)
	ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
	ON_COMMAND(ID_APP_HELP, OnAppHelp)

	//}}AFX_MSG_MAP
	// стандартний файл для команд
	ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
	ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
	// Стандартне розпечатування файлів установки
	ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CSIEM_AppApp construction
//Опис конструктора
CSIEM_AppApp::CSIEM_AppApp()
{
}

/////////////////////////////////////////////////////////////////////////////
CSIEM_AppApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CSIEM_AppApp initialization

//Ініціалізація вікна
BOOL CSIEM_AppApp::InitInstance()
{
	AfxEnableControlContainer();

	//Ініціалізація лінковки статичного управління MFC
#ifdef _AFXDLL
	Enable3dControls();
#else
	Enable3dControlsStatic();
#endif


	SetRegistryKey(_T("SIEM_App"));

  // Завантаження стандартних INI файлів настроювання (підключення MRU)
	LoadStdProfileSettings();

  //Створення  SDI фрейму вікна
	CSingleDocTemplate* pDocTemplate;
	pDocTemplate = new CSingleDocTemplate(
		IDR_MAINFRAME,
```

```
            RUNTIME_CLASS(CSIEM_AppDoc),
            RUNTIME_CLASS(CMainFrame),
            RUNTIME_CLASS(CSIEM_AppView));
      AddDocTemplate(pDocTemplate);

        // Аналіз командного рядка, DDE, відкриття файлу
      CCommandLineInfo cmdInfo;
      ParseCommandLine(cmdInfo);

        //Видалення параметрів командного рядка
      if (!ProcessShellCommand(cmdInfo))
            return FALSE;

        // Ініціалізація вікна, його відображення й відновлення
      m_pMainWnd->ShowWindow(SW_SHOW);
      m_pMainWnd->UpdateWindow();

      return TRUE;
}


//////////////////////////////////////////////////////////////////////////
// клас обробки й відображення вікна інформації про програму
class CAboutDlg : public CDialog
{
public:
  //Конструктор
      CAboutDlg();

// Діалогові дані
      //{{AFX_DATA(CAboutDlg)
  // Покажчик на ресурс оголошення діалогового вікна
      enum { IDD = IDD_ABOUTBOX };
      //}}AFX_DATA

            //{{AFX_VIRTUAL(CAboutDlg)
      protected:
    // DDX/DDV підтримка обміну даних
      virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV
супроводження
      //}}AFX_VIRTUAL

// Реалізація програми
protected:
      //{{AFX_MSG(CAboutDlg)
      //}}AFX_MSG
  //Декларація мапінгу повідомлень
      DECLARE_MESSAGE_MAP()
};

//Конструктор
CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
      //{{AFX_DATA_INIT(CAboutDlg)
      //}}AFX_DATA_INIT
}

//функція зчитування й установки даних вікна
void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
      CDialog::DoDataExchange(pDX);
      //{{AFX_DATA_MAP(CAboutDlg)
      //}}AFX_DATA_MAP
}
//Мапінг Windows подій, перехоплення пост повідомлень
BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
      //{{AFX_MSG_MAP(CAboutDlg)
      //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

```
/////////////////////////////////////////////////////////////////////////////
// CHelpDlg dialog used for App Help
// клас обробки й відображення вікна допомоги по програмі
class CHelpDlg : public CDialog
{
public:
  //Конструктор
      CHelpDlg();

// Діалогові дані
      //{{AFX_DATA(CHelpDlg)
  // Покажчик на ресурс оголошення діалогового вікна
      enum { IDD = IDD_HELPBOX };
      //}}AFX_DATA

            //{{AFX_VIRTUAL(CHelpDlg)
      protected:
    // DDX/DDV підтримка обміну даних
      virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
      //}}AFX_VIRTUAL

// Implementation
protected:
      //{{AFX_MSG(CHelpDlg)
      //}}AFX_MSG
  //Декларація мапінгу повідомлень
      DECLARE_MESSAGE_MAP()
};

//Конструктор
CHelpDlg::CHelpDlg() : CDialog(CHelpDlg::IDD)
{
      //{{AFX_DATA_INIT(CHelpDlg)
      //}}AFX_DATA_INIT
}

//функція зчитування й установки дані вікна
void CHelpDlg::DoDataExchange(CDataExchange* pDX)
{
  FILE * f = NULL;
  if (fopen_s(&f, "help.txt", "r+t") == 0) {
#define BUFFER_SIZE 10240
    size_t count = 0;
    char buff[BUFFER_SIZE];
    char text[BUFFER_SIZE];
    count = fread(buff, sizeof( char ), BUFFER_SIZE, f);
    fclose(f);
    size_t index = 0;
    for (size_t i = 0 ; i < count; i++) {
      if (buff[i] == 0x0A) {
        text[index] = '\r';
        index++;
      }
      text[index] = buff[i];
      index++;
    }
    text[index] = 0;
    SetDlgItemText(IDC_HELP_TEXT, text);
  }

  CDialog::DoDataExchange(pDX);
      //{{AFX_DATA_MAP(CHelpDlg)
      //}}AFX_DATA_MAP
}

//Мапінг Windows подій, перехоплення пост повідомлень
```

```
BEGIN_MESSAGE_MAP(CHelpDlg, CDialog)
      //{{AFX_MSG_MAP(CHelpDlg)
      //}}AFX_MSG_MAP
END_MESSAGE_MAP()

//функція створення й відкриття діалогового вікна інформації про програму
void CSIEM_AppApp::OnAppAbout()
{
      CAboutDlg aboutDlg;
      aboutDlg.DoModal();
}

//функція створення й відкриття діалогового вікна допомоги по програмі
void CSIEM_AppApp::OnAppHelp()
{
      CHelpDlg helpDlg;
      helpDlg.DoModal();
}

/////////////////////////////////////////////////////////////////////////////
```

DefaultActionDlg.cpp - Підключення основних оголошень діалогового вікна

```
//Описувач класу програми
//Підключення основних оголошень діалогового вікна
#include "stdafx.h"
#include "SIEM_app.h"
#include "DefaultActionDlg.h"

// Ініціалізація дебаг інформації
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CDefaultActionDlg dialog


//Опис конструктора
CDefaultActionDlg::CDefaultActionDlg(CWnd* pParent /*=NULL*/)
      : CDialog(CDefaultActionDlg::IDD, pParent)
{
      //{{AFX_DATA_INIT(CDefaultActionDlg)
      //}}AFX_DATA_INIT

  //Завдання первісної основної дії
      action = PF_ACTION_FORWARD;
}

//функція зчитування й установки даних вікна
void CDefaultActionDlg::DoDataExchange(CDataExchange* pDX)
{
      CDialog::DoDataExchange(pDX);
      //{{AFX_DATA_MAP(CDefaultActionDlg)
            // NOTE: the ClassWizard will add DDX and DDV calls here
      //}}AFX_DATA_MAP
}

//Мапінг Windows подій, перехоплення пост повідомлень
BEGIN_MESSAGE_MAP(CDefaultActionDlg, CDialog)
      //{{AFX_MSG_MAP(CDefaultActionDlg)
      //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
//Ініціалізація вікна
BOOL CDefaultActionDlg::OnInitDialog()
{
      CDialog::OnInitDialog();
      //Установка значень управління вікна
      if(action == PF_ACTION_DROP)
            CheckRadioButton(IDC_RADIOFORWARD, IDC_RADIODROP, IDC_RADIODROP);

      else
            CheckRadioButton(IDC_RADIOFORWARD, IDC_RADIODROP, IDC_RADIOFORWARD);

      return TRUE;
}

//обробка подій (пост повідомлень) Windows
void CDefaultActionDlg::OnOK()
{
  //Зчитування значення
      int id = GetCheckedRadioButton(IDC_RADIOFORWARD, IDC_RADIODROP);

  //Збереження  поточної основної дії
      if(id == IDC_RADIODROP)
            action = PF_ACTION_DROP;
```

```
        else
              action = PF_ACTION_FORWARD;

    //Виклик оброблювача події предка для завершення коректної реакції на подію
        CDialog::OnOK();
}
```

SIEM_AppView.cpp – Формування вікон

```
//Описувач класу програми
//Підключення основних оголошень діалогового вікна
#include "stdafx.h"
#include "SIEM_App.h"

#include "SIEM_AppDoc.h"
#include "SIEM_AppView.h"
#include "SockUtil.h"
#include "PacketFilter.h"

// Ініціалізація дебаг інформації
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CSIEM_AppView
//Мапінг Windows подій, перехоплення пост повідомлень
IMPLEMENT_DYNCREATE(CSIEM_AppView, CFormView)

BEGIN_MESSAGE_MAP(CSIEM_AppView, CFormView)
	//{{AFX_MSG_MAP(CSIEM_AppView)

	//}}AFX_MSG_MAP
	// Standard printing commands
	ON_COMMAND(ID_FILE_PRINT, CFormView::OnFilePrint)
	ON_COMMAND(ID_FILE_PRINT_DIRECT, CFormView::OnFilePrint)
	ON_COMMAND(ID_FILE_PRINT_PREVIEW, CFormView::OnFilePrintPreview)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CSIEM_AppView construction/destruction
//Опис конструктора
CSIEM_AppView::CSIEM_AppView()
	: CFormView(CSIEM_AppView::IDD)
{
	//{{AFX_DATA_INIT(CSIEM_AppView)
	//}}AFX_DATA_INIT
	}

//Опис деструктора
CSIEM_AppView::~CSIEM_AppView()
{
}

//функція зчитування й установки дані вікна
void CSIEM_AppView::DoDataExchange(CDataExchange* pDX)
{
	CFormView::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CSIEM_AppView)
	DDX_Control(pDX, IDC_LIST1, m_rules);
	//}}AFX_DATA_MAP
}

//обробка повідомлення передстворення основного вікна з викликом оброблювача
предка для коректної обробки події
BOOL CSIEM_AppView::PreCreateWindow(CREATESTRUCT& cs)
{


	return CFormView::PreCreateWindow(cs);
}

//Ініціалізація вікна
```

```
void CSIEM_AppView::OnInitialUpdate()
{
	CFormView::OnInitialUpdate();
	GetParentFrame()->RecalcLayout();
	ResizeParentToFit();

	RECT rc;
	m_rules.GetClientRect(&rc);

	int width=rc.right-rc.left-110;
	m_rules.InsertColumn(0, "Source IP",LVCFMT_LEFT , width/6, 0);
	m_rules.InsertColumn(1, "Source Mask",LVCFMT_LEFT , width/6, 1);
	m_rules.InsertColumn(2, "Source Port",LVCFMT_LEFT ,width/6, 2);
	m_rules.InsertColumn(3, "Dest. IP",LVCFMT_LEFT , width/6, 3);
	m_rules.InsertColumn(4, "Dest. Mask",LVCFMT_LEFT , width/6, 4);
	m_rules.InsertColumn(5, "Dest. Port",LVCFMT_LEFT , width/6, 5);
	m_rules.InsertColumn(6, "Protocol",LVCFMT_LEFT ,60, 6);
	m_rules.InsertColumn(7, "Action",LVCFMT_LEFT , 50, 7);

	m_rules.SetExtendedStyle(LVS_EX_FULLROWSELECT | LVS_EX_GRIDLINES);


}

/////////////////////////////////////////////////////////////////////////////
// CSIEM_AppView printing
//Не використовується, але взагалі для друку , і виводу інформації на друк
BOOL CSIEM_AppView::OnPreparePrinting(CPrintInfo* pInfo)
{
	// default preparation
	return DoPreparePrinting(pInfo);
}
//Не використовується, але взагалі для друку , і виводу інформації на друк
void CSIEM_AppView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{

}
//Не використовується, але взагалі для друку , і виводу інформації на друк
void CSIEM_AppView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{

}
//Не використовується, але взагалі для друку , і виводу інформації на друк
void CSIEM_AppView::OnPrint(CDC* pDC, CPrintInfo* /*pInfo*/)
{
	//
}

/////////////////////////////////////////////////////////////////////////////
// CSIEM_AppView diagnostics
//опис функцій обробки дебаг інформації
#ifdef _DEBUG
void CSIEM_AppView::AssertValid() const
{
	CFormView::AssertValid();
}

void CSIEM_AppView::Dump(CDumpContext& dc) const
{
	CFormView::Dump(dc);
}

CSIEM_AppDoc* CSIEM_AppView::GetDocument() // non-debug version is inline
{
	ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CSIEM_AppDoc)));
	return (CSIEM_AppDoc*)m_pDocument;
}
#endif //_DEBUG
```

```
////////////////////////////////////////////////////////////////////////////
// CSIEM_AppView message handlers
//опис функції відновлення списку правил в інтерфейсі
void CSIEM_AppView::UpdateList()
{
	CSIEM_AppDoc *doc = GetDocument();

	int action = (doc->defaultAction == PF_ACTION_FORWARD) ? 1:0;

	// обновлення листа керування
	m_rules.DeleteAllItems();

	unsigned int i;
	for(i=0;i<doc->nRules;i++)
	{
		AddRuleToList(doc->rules[i].sourceIp,
				doc->rules[i].sourceMask,
				doc->rules[i].sourcePort,
				doc->rules[i].destinationIp,
				doc->rules[i].destinationMask,
				doc->rules[i].destinationPort,
				doc->rules[i].protocol,
				action);
	}
}

//опис функції додавання правила в інтерфейс еа відображення інформації про
правило
void CSIEM_AppView::AddRuleToList(unsigned long srcIp,
					unsigned long srcMask,
					unsigned short srcPort,
					unsigned long dstIp,
					unsigned long dstMask,
					unsigned short dstPort,
					unsigned int protocol,
					int action)
{
	char ip[16];
	char port[6];
	LVITEM it;
	int pos;


	it.mask		= LVIF_TEXT;
	it.iItem	= m_rules.GetItemCount();
	it.iSubItem = 0;
	it.pszText	= (srcIp == 0) ? "All" : IpToString(ip, srcIp);
	pos		= m_rules.InsertItem(&it);

	it.iItem	= pos;
	it.iSubItem = 1;
	it.pszText	= IpToString(ip, srcMask);
	m_rules.SetItem(&it);

	it.iItem	= pos;
	it.iSubItem = 2;

	if(protocol != ICMP_PROTOCOL)
		it.pszText	= (srcPort == 0) ? "All" : itoa(srcPort, port, 10);

	else
		it.pszText	= (srcPort == 255) ? "All" : itoa(srcPort, port, 10);

	m_rules.SetItem(&it);

	it.iItem	= pos;
	it.iSubItem = 3;
	it.pszText	= (dstIp == 0) ? "All" : IpToString(ip, dstIp);
	m_rules.SetItem(&it);
```

```
        it.iItem    = pos;
        it.iSubItem = 4;
        it.pszText  = IpToString(ip, dstMask);
        m_rules.SetItem(&it);

        it.iItem    = pos;
        it.iSubItem = 5;

        if(protocol != ICMP_PROTOCOL)
              it.pszText  = (dstPort == 0) ? "All" : itoa(dstPort, port, 10);

        else
              it.pszText  = (dstPort == 255) ? "All" : itoa(dstPort, port, 10);

        m_rules.SetItem(&it);


        it.iItem    = pos;
        it.iSubItem = 6;

        if(protocol == 1)
              it.pszText = "ICMP";

        else if(protocol == 6)
              it.pszText = "TCP";

        else if(protocol == 17)
              it.pszText = "UDP";

        else
              it.pszText = "All";

        m_rules.SetItem(&it);


        it.iItem    = pos;
        it.iSubItem = 7;
        it.pszText = action ? "Drop" : "Forward";
        m_rules.SetItem(&it);
}
```

MainFrm.cpp – Ініціалізація й створення головного вікна і його компонентів, основна робота програми

```
//Описувач класу програми
//Підключення основних оголошень діалогового вікна
#include "stdafx.h"
#include "SIEM_App.h"

#include "MainFrm.h"
#include "RuleDlg.h"
#include "DefaultActionDlg.h"
#include "SIEM_AppDoc.h"
#include "SIEM_AppView.h"
#include "SockUtil.h"
#include "rules.h"

// Ініціалізація дебаг інформації
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CMainFrame
//Мапінг Windows подій, перехоплення пост повідомлень
IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
	//{{AFX_MSG_MAP(CMainFrame)
	ON_WM_CREATE()
	ON_COMMAND(ID_BUTTONSTART, OnButtonstart)
	ON_COMMAND(ID_BUTTONADD, OnButtonadd)
	ON_COMMAND(ID_BUTTONDEL, OnButtondel)
	ON_COMMAND(ID_BUTTONSTOP, OnButtonstop)
	ON_UPDATE_COMMAND_UI(ID_BUTTONSTART, OnUpdateButtonstart)
	ON_UPDATE_COMMAND_UI(ID_BUTTONSTOP, OnUpdateButtonstop)
	ON_COMMAND(IDMENU_ADDRULE, OnMenuAddrule)
	ON_COMMAND(IDMENU_DELRULE, OnMenuDelrule)
	ON_COMMAND(ID_MENUSTART, OnMenustart)
	ON_UPDATE_COMMAND_UI(ID_MENUSTART, OnUpdateMenustart)
	ON_COMMAND(ID_MENUSTOP, OnMenustop)
	ON_UPDATE_COMMAND_UI(ID_MENUSTOP, OnUpdateMenustop)
	ON_COMMAND(ID_APP_EXIT, OnAppExit)
	ON_COMMAND(IDMENU_LOADRULES, OnLoadRules)
	ON_COMMAND(IDMENU_SAVERULES, OnSaveRules)
	ON_COMMAND(IDMENU_SETDEFAULT, OnMenuSetdefault)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

static UINT indicators[] =
{
	ID_SEPARATOR,
/*	ID_INDICATOR_CAPS,
	ID_INDICATOR_NUM,
	ID_INDICATOR_SCRL,
  */
};

/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction
//Опис конструктора
CMainFrame::CMainFrame()
{
	started = FALSE;
}
//Опис деструктора
CMainFrame::~CMainFrame()
{
```

```
}

//ініціалізація й створення вікна і його компонентів
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
  //створення вікна
      if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
            return -1;

  //створення ToolBar
      if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD | WS_VISIBLE |
CBRS_TOP
            | CBRS_GRIPPER | CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC) ||
            !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
      {
            TRACE0("Failed to create toolbar\n");
            return -1;      // помилка створення
      }

  //створення StatusBar
      if (!m_wndStatusBar.Create(this) ||
              !m_wndStatusBar.SetIndicators(indicators,
sizeof(indicators)/sizeof(UINT))
     )
      {
            TRACE0("Failed to create status bar\n");
            return -1;      // помилка створення
      }


      //m_wndToolBar.EnableDocking(CBRS_ALIGN_RIGHT);
  //EnableDocking(CBRS_ALIGN_ANY);
      //DockControlBar(&m_wndToolBar);

  //Установка заголовка
      this->SetWindowText("SIEM_");

      return 0;
}

//обробка повідомлення передстворення основного вікна з викликом оброблювача
предка для коректної обробки події
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
      if( !CFrameWnd::PreCreateWindow(cs) )
            return FALSE;


            cs.style &= ~ FWS_ADDTOTITLE;

      return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics
//опис функцій обробки дебаг інформації
#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
      CFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
      CFrameWnd::Dump(dc);
}

#endif //_DEBUG
```

```
//////////////////////////////////////////////////////////////////////////
// CMainFrame заголовок повідомлення

//Функція запуску програми
void CMainFrame::OnButtonstart()
{
	CSIEM_AppDoc *doc = (CSIEM_AppDoc *)GetActiveDocument();
	unsigned int i;
	DWORD result;
	PIP_ADAPTER_INFO pAdapterInfo = NULL, aux;
	IP_ADDR_STRING *localIp;
	unsigned long len = 0;



  //Пошук адаптера мережної карти
	GetAdaptersInfo(pAdapterInfo, &len);

	pAdapterInfo = (PIP_ADAPTER_INFO) malloc (len);

	result = GetAdaptersInfo(pAdapterInfo, &len);

	if(result != ERROR_SUCCESS)
	{
		AfxMessageBox("Error getting adapters info.");

		return;
	}

	// Посилка правил на інтерфейс адаптера
	for(i=0;i<doc->nRules;i++)
	{
		// на всі знайдені адаптери
		for(aux=pAdapterInfo;aux != NULL;aux=aux->Next)
		{
			// на кожний IP адаптера
			for(localIp=&aux->IpAddressList;localIp!=NULL;localIp=localIp-
>Next)
			{
				pckFilter.AddFilter(CharToIp(localIp->IpAddress.String),
                                                ANY_DIRECTION,
                                                doc->rules[i].sourceIp,
                                                doc->rules[i].sourceMask,
                                                doc-
>rules[i].destinationIp,
                                                doc-
>rules[i].destinationMask,
                                                doc->rules[i].sourcePort,
                                                doc-
>rules[i].destinationPort,
                                                doc->rules[i].protocol);

			}
		}

	}

	started = TRUE;
}

//функція вимикання програми
void CMainFrame::OnButtonstop()
{
	pckFilter.RemoveAll();

	started = FALSE;
}
```

```
//функція додавання правила
void CMainFrame::OnButtonadd()
{
	CSIEM_AppDoc *doc = (CSIEM_AppDoc *)GetActiveDocument();
	CRuleDlg dlg;

	// перевірка правильності номерів
	if(doc->nRules < MAX_RULES )
	{
		dlg.defaultAction = pckFilter.GetDefaultAction();

		if(dlg.DoModal() == IDOK)
		{
			// додавання правильного номера до листа правильних адрес

			if(doc->AddRule(dlg.srcIp, dlg.srcMask, dlg.srcPort,
dlg.dstIp, dlg.dstMask, dlg.dstPort, dlg.protocol, dlg.cAction) != 0)
				AfxMessageBox("Error adding the rule.");

			else
			{
				// Після цього коректуються правила
				CSIEM_AppView *view = (CSIEM_AppView *)GetActiveView();

				view->UpdateList();
			}

		}
	}

	else
		AfxMessageBox("You can't add more rules.");
}

//функція видалення правила
void CMainFrame::OnButtondel()
{
	CSIEM_AppView *view = (CSIEM_AppView *)GetActiveView();


		POSITION pos = view->m_rules.GetFirstSelectedItemPosition();
	if (pos == NULL)
	{
		AfxMessageBox("Select a Rule, please.");

		return;
	}

	int position;
	position = view->m_rules.GetNextSelectedItem(pos);

	CSIEM_AppDoc *doc = (CSIEM_AppDoc *)GetActiveDocument();
	doc->DeleteRule(position);


	// перегляд змін в правилах
	view->UpdateList();

}

//функція перемикання стану меню по запуску програми
void CMainFrame::OnUpdateButtonstart(CCmdUI* pCmdUI)
{
	if(started)
		pCmdUI->Enable(FALSE);

	else
		pCmdUI->Enable(TRUE);
}
```

```
//функція перемикання стану меню по зупинці програми
void CMainFrame::OnUpdateButtonstop(CCmdUI* pCmdUI)
{
	if(started)
		pCmdUI->Enable(TRUE);

	else
		pCmdUI->Enable(FALSE);
}

//функція обробки меню по додаванню правила
void CMainFrame::OnMenuAddrule()
{
	OnButtonadd();
}

//функція обробки меню по видаленню правила
void CMainFrame::OnMenuDelrule()
{
	OnButtondel();
}


//функція обробки меню по старту програми
void CMainFrame::OnMenustart()
{
	OnButtonstart();
}

//функція обробки відновлення меню
void CMainFrame::OnUpdateMenustart(CCmdUI* pCmdUI)
{
	if(started)
		pCmdUI->Enable(FALSE);

	else
		pCmdUI->Enable(TRUE);

}

//функція обробки меню по зупинці програми
void CMainFrame::OnMenustop()
{
	OnButtonstop();
}

//функція перемикання стану головного меню по зупинці програми
void CMainFrame::OnUpdateMenustop(CCmdUI* pCmdUI)
{
	if(started)
		pCmdUI->Enable(TRUE);

	else
		pCmdUI->Enable(FALSE);

}

//функція обробки виходу із програми
void CMainFrame::OnAppExit()
{

}

//функція обробки завантаження правил з файлу
void CMainFrame::OnLoadRules()
{
	CFile file;
	CFileException e;
```

```
	DWORD nRead;

	CSIEM_AppDoc *doc = (CSIEM_AppDoc *)GetActiveDocument();

	CFileDialog dg(TRUE,NULL, NULL, OFN_HIDEREADONLY | OFN_CREATEPROMPT,"Rule
Files(*.rul)|*.rul|all(*.*)|*.*||", NULL);
	if(dg.DoModal()==IDCANCEL)
		return;

	CString nf=dg.GetPathName();

	if(nf.GetLength() == 0)
	{
		AfxMessageBox("This file name isn't valid.");

		return;
	}

	if( !file.Open(nf, CFile::modeRead, &e ) )
	{
		AfxMessageBox("Error openning the file.");

		return;
	}

	doc->ResetRules();

	PFFORWARD_ACTION action;
	file.Read(&action, sizeof(PFFORWARD_ACTION));

	if(action != pckFilter.GetDefaultAction())
	{
		pckFilter.RemoveAll();

		pckFilter.SetDefaultAction(action);
		doc->defaultAction = action;
	}



	RuleInfo rule;

    do
    {
		nRead = file.Read(&rule, sizeof(RuleInfo));

		if(nRead == 0)
			break;

		if(doc->AddRule(rule.sourceIp,
                        rule.sourceMask,
                            rule.sourcePort,
                            rule.destinationIp,
                            rule.destinationMask,
                            rule.destinationPort,
                            rule.protocol,
                            1)  != 0)
		{

			AfxMessageBox("Error adding a rule.");

			break;
		}

    }while (1);
```

```
	CSIEM_AppView *view = (CSIEM_AppView *)GetActiveView();

	view->UpdateList();
}

//функція обробки збереження правил у файл
void CMainFrame::OnSaveRules()
{
	CSIEM_AppDoc *doc = (CSIEM_AppDoc *)GetActiveDocument();

	if(doc->nRules == 0)
	{
		AfxMessageBox("There isnt Rules to Save.");

		return;
	}


	CFileDialog dg(FALSE, NULL, NULL, OFN_HIDEREADONLY |
OFN_CREATEPROMPT,"Rule Files(*.rul)|*.rul|all(*.*)|*.*||", NULL);
	if(dg.DoModal()==IDCANCEL)
		return;

	CString nf=dg.GetPathName();

	if(nf.GetLength() == 0)
	{
		AfxMessageBox("This file name isn't valid.");

		return;
	}

	CFile file;
	CFileException e;

	if( !file.Open( nf, CFile::modeCreate | CFile::modeWrite, &e ) )
	{
		AfxMessageBox("Error openning the file.");

		return;
	}


	PFFORWARD_ACTION action = pckFilter.GetDefaultAction();
	file.Write(&action, sizeof(PFFORWARD_ACTION));

	unsigned int i;

		for(i=0;i<doc->nRules;i++)
	{
		file.Write(&doc->rules[i], sizeof(RuleInfo));
	}

	file.Close();
}

//скидання даних і реініціалізації програми у вихідне положення
void CMainFrame::OnMenuSetdefault()
{
	CDefaultActionDlg dlg;

	dlg.action = pckFilter.GetDefaultAction();

	if(dlg.DoModal() == IDOK)
	{
		if(dlg.action != pckFilter.GetDefaultAction())
		{
			pckFilter.RemoveAll();
```

```
                    pckFilter.SetDefaultAction(dlg.action);

                    CSIEM_AppDoc *doc = (CSIEM_AppDoc *)GetActiveDocument();
                    doc->defaultAction = dlg.action;

                    CSIEM_AppView *view = (CSIEM_AppView *)GetActiveView();
                    view->UpdateList();
                }
            }
        }
```

PacketFilter.cpp – Формування правил фільтру

```
//Описувач класу програми
//Підключення основних оголошень діалогового вікна
#include "stdafx.h"
#include "PacketFilter.h"


#include <stdlib.h>
#include <stdio.h>



/*********************************************
Class CPacketFilter.
*********************************************/
//Опис конструктора
CPacketFilter::CPacketFilter()
{
      defaultAction = PF_ACTION_FORWARD;
}
//Опис деструктора
CPacketFilter::~CPacketFilter()
{
      RemoveAll();
}

//Опис функції видалення всіх фільтрів з інтерфейсу
void CPacketFilter::RemoveAll()
{
      POSITION pos = interfacesTable.GetStartPosition();

      CPacketFilterInterface pckInt;
      unsigned long ip;

      while( pos != NULL )
      {

            interfacesTable.GetNextAssoc(pos, ip, pckInt);

            PfDeleteInterface(pckInt.hInterface);

      }

      interfacesTable.RemoveAll();
}


//Опис функції додавання фільтра в інтерфейс
int CPacketFilter::AddFilter(char   *localInterfaceIp,
                                          int  direction,
                                          char *srcIp,
                                          char *srcMask,
                                          char *dstIp,
                                          char *dstMask,
                                          int  srcPort,
                                          int  dstPort,
                                          int  protocol)
{
      CPacketFilterInterface interfaceHandle;

            if(!interfacesTable.Lookup(CharToIp(localInterfaceIp),
interfaceHandle))
      {

            interfaceHandle.Create(CharToIp(localInterfaceIp), defaultAction);

            interfacesTable[CharToIp(localInterfaceIp)] = interfaceHandle;
```

```
	}


	// Заповнюю структуру фільтру
	PF_FILTER_DESCRIPTOR ipFlt;
	ipFlt.dwFilterFlags     = FD_FLAGS_NOSYN;
	ipFlt.dwRule            = 0;
	ipFlt.pfatType          = PF_IPV4;
	ipFlt.dwProtocol        = protocol;
	ipFlt.fLateBound        = 0;
	ipFlt.wSrcPort          = srcPort;
	ipFlt.wSrcPortHighRange = srcPort;
	ipFlt.wDstPort          = dstPort;
	ipFlt.wDstPortHighRange = dstPort;

	unsigned long lIpSrc    = CharToIp(srcIp);
	unsigned long lIpDst    = CharToIp(dstIp);
	unsigned long lMaskSrc  = CharToIp(srcMask);
	unsigned long lMaskDst  = CharToIp(dstMask);

	ipFlt.SrcAddr = (PBYTE) &lIpSrc;
	ipFlt.SrcMask = (PBYTE) &lMaskSrc;
	ipFlt.DstAddr = (PBYTE) &lIpDst;
	ipFlt.DstMask = (PBYTE) &lMaskDst;

	DWORD errorCode;

	if(direction == IN_DIRECTION || direction == ANY_DIRECTION)
		errorCode = PfAddFiltersToInterface(interfaceHandle.hInterface,
                                                    1,
                                                    &ipFlt,
                                                    0,
                                                    NULL,
                                                    NULL);

	if(direction == OUT_DIRECTION || direction == ANY_DIRECTION)
		errorCode = PfAddFiltersToInterface(interfaceHandle.hInterface,
                                                    0,
                                                    NULL,
                                                    1,
                                                    &ipFlt,
                                                    NULL);

	else
		return -2;

	if(errorCode != NO_ERROR)
		return -1;

	return 0;
}


//Опис функції додавання фільтра в інтерфейс по іншому набору параметрів
int CPacketFilter::AddFilter(unsigned long  localInterfaceIp,
                                        int             direction,
                                        unsigned long   srcIp,
                                        unsigned long   srcMask,
                                        unsigned long   dstIp,
                                        unsigned long   dstMask,
                                        int  srcPort,
                                        int  dstPort,
                                        int  protocol)
{
	CPacketFilterInterface interfaceHandle;

		if(!interfacesTable.Lookup(localInterfaceIp, interfaceHandle))
	{
```

```
            interfaceHandle.Create(localInterfaceIp, defaultAction);

            interfacesTable[localInterfaceIp] = interfaceHandle;
      }



      PF_FILTER_DESCRIPTOR ipFlt;
      ipFlt.dwFilterFlags     = 0;
      ipFlt.dwRule            = 0;
      ipFlt.pfatType          = PF_IPV4;
      ipFlt.dwProtocol        = protocol;
      ipFlt.fLateBound        = 0;
      ipFlt.wSrcPort          = srcPort;
      ipFlt.wSrcPortHighRange = srcPort;
      ipFlt.wDstPort          = dstPort;
      ipFlt.wDstPortHighRange = dstPort;


      ipFlt.SrcAddr = (PBYTE) &srcIp;
      ipFlt.SrcMask = (PBYTE) &srcMask;
      ipFlt.DstAddr = (PBYTE) &dstIp;
      ipFlt.DstMask = (PBYTE) &dstMask;

      DWORD errorCode;


      if(direction == IN_DIRECTION || direction == ANY_DIRECTION)
            errorCode = PfAddFiltersToInterface(interfaceHandle.hInterface,
                                                1,
                                                &ipFlt,
                                                0,
                                                NULL,
                                                NULL);

      if(direction == OUT_DIRECTION || direction == ANY_DIRECTION)
            errorCode = PfAddFiltersToInterface(interfaceHandle.hInterface,
                                                0,
                                                NULL,
                                                1,
                                                &ipFlt,
                                                NULL);

      else
            return -2;

      if(errorCode != NO_ERROR)
            return -1;

      return 0;
}

//Опис функції видалення зазначеного фильтру з інтерфейсу
int CPacketFilter::RemoveFilter(char     *localInterfaceIp,
                                int         direction,
                                char  *srcIp,
                                char  *srcMask,
                                char  *dstIp,
                                char  *dstMask,
                                int         srcPort,
                                int         dstPort,
                                int         protocol)
{
      CPacketFilterInterface interfaceHandle;

            if(!interfacesTable.Lookup(CharToIp(localInterfaceIp),
interfaceHandle))
      {
            return -1;
```

```
	}


	// Заповнюю структуру фільтру
	PF_FILTER_DESCRIPTOR ipFlt;
	ipFlt.dwFilterFlags     = FD_FLAGS_NOSYN;
	ipFlt.dwRule            = 0;
	ipFlt.pfatType          = PF_IPV4;
	ipFlt.dwProtocol        = protocol;
	ipFlt.fLateBound        = 0;
	ipFlt.wSrcPort          = srcPort;
	ipFlt.wSrcPortHighRange = srcPort;
	ipFlt.wDstPort          = dstPort;
	ipFlt.wDstPortHighRange = dstPort;

	unsigned long lIpSrc    = CharToIp(srcIp);
	unsigned long lIpDst    = CharToIp(dstIp);
	unsigned long lMaskSrc  = CharToIp(srcMask);
	unsigned long lMaskDst  = CharToIp(dstMask);

	ipFlt.SrcAddr = (PBYTE) &lIpSrc;
	ipFlt.SrcMask = (PBYTE) &lMaskSrc;
	ipFlt.DstAddr = (PBYTE) &lIpDst;
	ipFlt.DstMask = (PBYTE) &lMaskDst;

	DWORD errorCode;

		if(direction == IN_DIRECTION || direction == ANY_DIRECTION)
		errorCode = PfRemoveFiltersFromInterface(interfaceHandle.hInterface,
                                                         1,
                                                         &ipFlt,
                                                         0,
                                                         NULL);


	if(direction == OUT_DIRECTION || direction == ANY_DIRECTION)
		errorCode = PfRemoveFiltersFromInterface(interfaceHandle.hInterface,
                                                         0,
                                                         NULL,
                                                         1,
                                                         &ipFlt);

	else
		return -2;

	if(errorCode != NO_ERROR)
		return -1;

	return 0;
}


//Опис функції видалення зазначеного фильтру з інтерфейсу по іншому наборі
параметрів
int CPacketFilter::RemoveFilter(unsigned long  localInterfaceIp,
                                               int
	direction,
                                               unsigned long     srcIp,

                                               unsigned long     srcMask,
                                               unsigned long     dstIp,
                                               unsigned long     dstMask,
                                               int                    srcPort,
                                               int                    dstPort,
                                               int                    protocol)
{
	CPacketFilterInterface interfaceHandle;

	// Якщо інтерфейс не створенй, то створюю його.
	if(!interfacesTable.Lookup(localInterfaceIp, interfaceHandle))
```

```
	{
		return -1;
	}


	// Заповнюю структуру фільтру
	PF_FILTER_DESCRIPTOR ipFlt;
	ipFlt.dwFilterFlags     = FD_FLAGS_NOSYN;
	ipFlt.dwRule            = 0;
	ipFlt.pfatType          = PF_IPV4;
	ipFlt.dwProtocol        = protocol;
	ipFlt.fLateBound        = 0;
	ipFlt.wSrcPort          = srcPort;
	ipFlt.wSrcPortHighRange = srcPort;
	ipFlt.wDstPort          = dstPort;
	ipFlt.wDstPortHighRange = dstPort;

	ipFlt.SrcAddr = (PBYTE) &srcIp;
	ipFlt.SrcMask = (PBYTE) &srcMask;
	ipFlt.DstAddr = (PBYTE) &dstIp;
	ipFlt.DstMask = (PBYTE) &dstMask;

	DWORD errorCode;

	// Додаю фільтр
	if(direction == IN_DIRECTION || direction == ANY_DIRECTION)
		errorCode = PfRemoveFiltersFromInterface(interfaceHandle.hInterface,
                                                 1,
                                                 &ipFlt,
                                                 0,
                                                 NULL);


	if(direction == OUT_DIRECTION || direction == ANY_DIRECTION)
		errorCode = PfRemoveFiltersFromInterface(interfaceHandle.hInterface,
                                                 0,
                                                 NULL,
                                                 1,
                                                 &ipFlt);


	else
		return -2;

	if(errorCode != NO_ERROR)
		return -1;

	return 0;
}


//Опис функції додавання глобального фільтра інтерфейсу
int CPacketFilter::AddGlobalFilter(char   *localInterfaceIp,
                                   int globalFilter)

{
	CPacketFilterInterface interfaceHandle;

	// Якщо інтерфейс не створений, то створюю його.
	if(!interfacesTable.Lookup(CharToIp(localInterfaceIp), interfaceHandle))
	{

		interfaceHandle.Create(CharToIp(localInterfaceIp), defaultAction);

		interfacesTable[CharToIp(localInterfaceIp)] = interfaceHandle;
	}


	DWORD errorCode;

	// Додаю глобальний фільтр
```

```
	errorCode = PfAddGlobalFilterToInterface(interfaceHandle.hInterface,
(GLOBAL_FILTER)globalFilter);

	if(errorCode != NO_ERROR)
		return -1;

	return 0;
}


//Опис функції видалення глобального фільтра інтерфейсу
int CPacketFilter::RemoveGlobalFilter(char	*localInterfaceIp,
						int	globalFilter)

{
	CPacketFilterInterface interfaceHandle;

	// Якщо інтерфейс не створенй, то створюю його.
	if(!interfacesTable.Lookup(CharToIp(localInterfaceIp), interfaceHandle))
	{
		return -1;
	}


	DWORD errorCode;

	// Додаю глобальний фільтр
	errorCode = PfRemoveGlobalFilterFromInterface(interfaceHandle.hInterface,
(GLOBAL_FILTER)globalFilter);

	if(errorCode != NO_ERROR)
		return -1;

	return 0;
}

//установка основної дії інтерфейсу
void CPacketFilter::SetDefaultAction(PFFORWARD_ACTION action)
{
	defaultAction = action;
}

//зчитування основної дії інтерфейсу
PFFORWARD_ACTION  CPacketFilter::GetDefaultAction()
{
	return defaultAction;
}

/*******************************************
Class CPacketFilterInterface.
*******************************************/
//інтерфейсний клас мережного адаптера
CPacketFilterInterface::CPacketFilterInterface()
{

}

CPacketFilterInterface::~CPacketFilterInterface()
{
//	PfDeleteInterface(hInterface);
}

// Створення інтерфейсу й асоціація його з IP
int CPacketFilterInterface::Create(unsigned long ip, PFFORWARD_ACTION
defaultAction)
{

	// Створення Інтерфейсу  й завдання початкової дії пропускати всі пакети
	DWORD errorCode = PfCreateInterface(0,
```

```
                                                defaultAction,
                                                defaultAction,
                                                FALSE,
                                                TRUE,
                                                &hInterface);

        if(errorCode != NO_ERROR)
        {
                return -1;
        }


        // Асоціація IP с інтерфейсом
        PBYTE lIp = (PBYTE)&ip;
        errorCode = PfBindInterfaceToIPAddress(hInterface, PF_IPV4, lIp);

        if(errorCode != NO_ERROR)
        {
                PfDeleteInterface(hInterface);

                hInterface = NULL;

                return -2;
        }

        return 0;
}

// перетворення строкового значення IP у цифрове представлення
unsigned long CharToIp(const char *sIp)
{
        int octets[4];
        int i;
        const char * auxCad = sIp;
        unsigned long lIp = 0;

        for(i = 0; i < 4; i++)
        {
                octets[i] = atoi(auxCad);

                if(octets[i] < 0 || octets[i] > 255)
                        return 0;

                lIp |= (octets[i] << (i*8));

                auxCad = strchr(auxCad, '.');

                if(auxCad == NULL && i!=3)
                        return -1;

                auxCad++;
        }


        return lIp;
}
```

ResizeDlg.cpp – Зміна розмірів вікон

```
**/
#include "stdafx.h"
#include "ResizeDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

#ifndef OBM_SIZE
#define OBM_SIZE 32766
#endif

CItemCtrl::CItemCtrl()
{
      m_nID = 0;
      m_stxLeft = CST_NONE;
      m_stxRight = CST_NONE;
      m_styTop = CST_NONE;
      m_styBottom = CST_NONE;
      m_bFlickerFree = 1;
      m_xRatio = m_cxRatio = 1.0;
      m_yRatio = m_cyRatio = 1.0;
}

CItemCtrl::CItemCtrl(const CItemCtrl& src)
{
      Assign(src);
}

CItemCtrl& CItemCtrl::operator=(const CItemCtrl& src)
{
      Assign(src);
      return *this;
}

void CItemCtrl::Assign(const CItemCtrl& src)
{
      m_nID = src.m_nID;
      m_stxLeft = src.m_stxLeft;
      m_stxRight = src.m_stxRight;
      m_styTop = src.m_styTop;
      m_styBottom = src.m_styBottom;
      m_bFlickerFree = src.m_bFlickerFree;
      m_bInvalidate = src.m_bInvalidate;
      m_wRect = src.m_wRect;
      m_xRatio = src.m_xRatio;
      m_cxRatio = src.m_cxRatio;
      m_yRatio = src.m_yRatio;
      m_cyRatio = src.m_cyRatio;
}

HDWP CItemCtrl::OnSize(HDWP hDWP,int sizeType, CRect *pnRect, CRect *poRect,
CRect *p0, CWnd *pDlg)
{
      CRect ctrlRect = m_wRect, curRect;
      CWnd   *pWnd = pDlg->GetDlgItem(m_nID);
      int         delta = pnRect->Width() - poRect->Width();
      int         delta = pnRect->Height()- poRect->Height();
      int       delta0 = pnRect->Width() - p 0-0->Width();
      int         delta0 = pnRect->Height()- p 0-0->Height();
      int         newCx, newCy;
      int         st, bUpdateRect = 1;
```

```
	// зміна зліва-горизонитально
	st = CST_NONE;
	if ((sizeType & WST_LEFT) && m_stxLeft != CST_NONE) {
		ASSERT((sizeType & WST_RIGHT) == 0);

		st = m_stxLeft;
	}
	else if ((sizeType & WST_RIGHT) && m_stxRight != CST_NONE) {
		ASSERT((sizeType & WST_LEFT) == 0);

		st = m_stxRight;
	}

	switch(st) {
	case CST_NONE:
		if (m_stxLeft == CST_ZOOM || m_stxRight == CST_ZOOM ||
				m_stxLeft == CST_DELTA_ZOOM || m_stxRight ==
CST_DELTA_ZOOM)
		{
			pWnd->GetWindowRect(&curRect);
			pDlg->ScreenToClient(&curRect);

			ctrlRect.left = curRect.left;
			ctrlRect.right = curRect.right;

			bUpdateRect = 0;
		}

		break;

	case CST_RESIZE:
		ctrlRect.right += delta;
		break;

	case CST_REPOS:
		ctrlRect.left += delta;
		ctrlRect.right += delta;
		break;

	case CST_RELATIVE:
		newCx = ctrlRect.Width();
		ctrlRect.left = (int)((double)m_xRatio * 1.0 * pnRect->Width() -
newCx / 2.0);
		ctrlRect.right = ctrlRect.left + newCx; /* (int)((double)m_xRatio *
1.0 * pnRect->Width() + newCx / 2.0); */
		break;

	case CST_ZOOM:
		ctrlRect.left = (int)(1.0 * ctrlRect.left  * (double)pnRect->Width()
/ p 0-0->Width());
		ctrlRect.right = (int)(1.0 * ctrlRect.right * (double)pnRect-
>Width() / p 0-0->Width());
		bUpdateRect = 0;
		break;

	case CST_DELTA_ZOOM:
		newCx = ctrlRect.Width();
		ctrlRect.right = (int)(ctrlRect.left * 1.0 + delta0 * 1.0 * m_xRatio
+ newCx * 1.0 + delta0 * m_cxRatio);
		ctrlRect.left += (int)(delta0 * 1.0 * m_xRatio);
		bUpdateRect = 0;
		break;

	default:
		break;
	}

	// зміна згори
	st = CST_NONE;
```

```
	if ((sizeType & WST_TOP) && (m_styTop != CST_NONE)) {
		ASSERT((sizeType & WST_BOTTOM) == 0);

		st = m_styTop;
	}
	else if ((sizeType & WST_BOTTOM) && (m_styBottom != CST_NONE)) {
		ASSERT((sizeType & WST_TOP) == 0);

		st = m_styBottom;
	}

	switch (st) {
	case CST_NONE:
		if (m_styTop == CST_ZOOM || m_styTop == CST_ZOOM ||
				m_styBottom == CST_DELTA_ZOOM || m_styBottom ==
CST_DELTA_ZOOM)
		{
			pWnd->GetWindowRect(&curRect);
			pDlg->ScreenToClient(&curRect);

			ctrlRect.top = curRect.top;
			ctrlRect.bottom = curRect.bottom;

			bUpdateRect = 0;
		}

		break;

	case CST_RESIZE:
		ctrlRect.bottom += delta;
		break;

	case CST_REPOS:
		ctrlRect.top += delta;
		ctrlRect.bottom += delta;
		break;

	case CST_RELATIVE:
		newCy = ctrlRect.Width();
		ctrlRect.top = (int)((double)m_yRatio * 1.0 * pnRect->Width() -
newCy / 2.0);
		ctrlRect.bottom = ctrlRect.top + newCy; /* (int)((double)m_yRatio *
1.0 * pnRect->Width() + newCy / 2.0); */

	case CST_ZOOM:
		ctrlRect.top = (int)(1.0 * ctrlRect.top * (double)pnRect->Height() /
p 0-0->Height());
		ctrlRect.bottom = (int)(1.0 * ctrlRect.bottom * (double)pnRect-
>Height() / p 0-0->Height());
		bUpdateRect = 0;
		break;

	case CST_DELTA_ZOOM:
		newCy = ctrlRect.Height();
		ctrlRect.bottom = (int)(ctrlRect.top * 1.0 + delta0 * 1.0 * m_yRatio
+ newCy * 1.0 + delta0 * m_cyRatio);
		ctrlRect.top += (int)(delta0 * 1.0 * m_yRatio);
		bUpdateRect = 0;
		break;

	default:
		break;
	}

	if (!bUpdateRect) {
		pWnd->GetWindowRect(&curRect);
		pDlg->ScreenToClient(&curRect);
	}
	else {
```

```
            curRect = m_wRect;
      }

      if (ctrlRect != curRect) {
            if (m_bInvalidate) {
                  pWnd->Invalidate();
            }

            hDWP = ::DeferWindowPos(hDWP, (HWND)*pWnd, NULL, ctrlRect.left,
ctrlRect.top, ctrlRect.Width(), ctrlRect.Height(), SWP_NOZORDER);

#if 1 /* why No effect!!!! */
            if (m_bInvalidate) {
                  pDlg->InvalidateRect(&curRect);
                  pDlg->UpdateWindow();
            }
#endif      /* No effect???? */

            if (bUpdateRect) {
                  m_wRect = ctrlRect;
            }
      }

      return hDWP;
}

IMPLEMENT_DYNAMIC(CResizeDlg, CDialog)
BEGIN_MESSAGE_MAP(CResizeDlg, CDialog)
      ON_WM_SIZING()
      ON_WM_SIZE()
      ON_WM_GETMINMAXINFO()
      ON_WM_ERASEBKGND()
END_MESSAGE_MAP()

CResizeDlg::CResizeDlg(const UINT resID,CWnd *pParent)
        : CDialog(resID,pParent)
{
      m_xSt = CST_RESIZE;
      m_xSt = CST_RESIZE;
      m_xMin = 32;
      m_yMin = 32;
      m_nDelaySide = 0;
}

CResizeDlg::~CResizeDlg()
{
      }

BOOL CResizeDlg::OnInitDialog()
{
      BOOL bret = CDialog::OnInitDialog();

      CRect cltRect;
      CBitmap cBmpSize;
      BITMAP      Bitmap;

      GetClientRect(&cltRect);
      m_clt0 = cltRect;
      ClientToScreen(&m_clt0);
      m_cltRect = m_clt0;

      cBmpSize.LoadOEMBitmap(OBM_SIZE);
      cBmpSize.GetBitmap(&Bitmap);

      m_wndSizeIcon.Create( NULL,
                        WS_CHILD | WS_VISIBLE | SS_BITMAP,
                        CRect(0, 0, Bitmap.bmWidth, Bitmap.bmHeight),
                        this, m_idSizeIcon );
      m_wndSizeIcon.SetBitmap(cBmpSize);
```

```
	m_wndSizeIcon.SetWindowPos(&wndTop,
				cltRect.right - Bitmap.bmWidth, cltRect.bottom -
Bitmap.bmHeight,
				0, 0,
				SWP_NOSIZE );
#if 0
	CRgn  cRgn;
	POINT bmpPt[3] = { { cltRect.right - Bitmap.bmWidth, cltRect.bottom },
					{ cltRect.right, cltRect.bottom -
Bitmap.bmHeight},
					{ cltRect.right, cltRect.bottom } };
	cRgn.CreatePolygonRgn(bmpPt, 3, WINDING);
	m_wndSizeIcon.SetWindowRgn(cRgn, TRUE);

	cRgn.Detach();
#endif

	cBmpSize.Detach();

	AddControl(m_idSizeIcon, CST_REPOS, CST_REPOS, CST_REPOS, CST_REPOS);

	CRect wRect;
	GetWindowRect(&wRect);
	m_xMin = wRect.Width();		// вбудована межа x
	m_yMin = wRect.Height();	// вбудована межа y

	return bret;
}

void CResizeDlg::OnSizing(UINT nSide, LPRECT lpRect)
{
	CDialog::OnSizing(nSide, lpRect);

	m_nDelaySide = nSide;
}

void CResizeDlg::OnSize(UINT nType,int cx,int cy)
{
	int    nCount;

	std::vector<CItemCtrl>::iterator it;

	CDialog::OnSize(nType,cx,cy);

	if((nCount = m_Items.size()) > 0) {
		CRect  cltRect;
		GetClientRect(&cltRect);
		ClientToScreen(cltRect);

		HDWP   hDWP;
		int    sizeType = WST_NONE;

#if 0
		int    delta = cltRect.Width() - m_cltRect.Width();
		int      delta = cltRect.Height()- m_cltRect.Height();
		int      mid = (cltRect.left + cltRect.right) / 2;
		int    mid = (cltRect.top + cltRect.bottom) / 2;
		CPoint csrPt(::GetMessagePos());

		if (delta) {
			if (csrPt.x < mid)
				sizeType |= WST_LEFT;
			else
				sizeType |= WST_RIGHT;
		}

		if (delta) {
			if (csrPt.y < mid)
				sizeType |= WST_TOP;
```

```
                  else
                        sizeType |= WST_BOTTOM;
            }
#else
            switch (m_nDelaySide) {
            case WMSZ_BOTTOM:
                  sizeType = WST_BOTTOM;
                  break;
            case WMSZ_BOTTOMLEFT:
                  sizeType = WST_BOTTOM|WST_LEFT;
                  break;
            case WMSZ_BOTTOMRIGHT:
                  sizeType = WST_BOTTOM|WST_RIGHT;
                  break;
            case WMSZ_LEFT:
                  sizeType = WST_LEFT;
                  break;
            case WMSZ_RIGHT:
                  sizeType = WST_RIGHT;
                  break;
            case WMSZ_TOP:
                  sizeType = WST_TOP;
                  break;
            case WMSZ_TOPLEFT:
                  sizeType = WST_TOP|WST_LEFT;
                  break;
            case WMSZ_TOPRIGHT:
                  sizeType = WST_TOP|WST_RIGHT;
                  break;
            default:
                  break;
            }
#endif

            if (sizeType != WST_NONE) {
                  hDWP = ::BeginDeferWindowPos(nCount);

                  for (it = m_Items.begin(); it != m_Items.end(); it++)
                        hDWP = it->OnSize(hDWP, sizeType, &cltRect, &m_cltRect,
&m_clt0, this);

                  ::EndDeferWindowPos(hDWP);
            }

            m_cltRect = cltRect;
      }

      m_nDelaySide = 0;
}

void CResizeDlg::OnGetMinMaxInfo(MINMAXINFO *pmmi)
{
      if ((HWND)m_wndSizeIcon == NULL)
            return;

      pmmi->ptMinTrackSize.x = m_xMin;
      pmmi->ptMinTrackSize.y = m_yMin;

      if (m_xSt == CST_NONE)
            pmmi->ptMaxTrackSize.x = pmmi->ptMaxSize.x = m_xMin;
      if (m_ySt == CST_NONE)
            pmmi->ptMaxTrackSize.y = pmmi->ptMaxSize.y = m_yMin;
}

BOOL CResizeDlg::OnEraseBkgnd(CDC *pDC)
{
      if (!(GetStyle() & WS_CLIPCHILDREN)) {
            std::vector<CItemCtrl>::const_iterator it;
```

```
            for(it = m_Items.begin();it != m_Items.end(); it++) {
                  // пропускається зміна іконки, якщо він скритий
                  if(it->m_nID == m_idSizeIcon &&
!m_wndSizeIcon.IsWindowVisible())
                        continue;

                  if(it->m_bFlickerFree && ::IsWindowVisible(GetDlgItem(it-
>m_nID)->GetSafeHwnd())) {
                        pDC->ExcludeClipRect(&it->m_wRect);
                  }
            }
      }

      CDialog::OnEraseBkgnd(pDC);
      return FALSE;
}

void CResizeDlg::AddControl( UINT nID,int xl, int xr, int yt, int yb, int
bFlickerFree,
                                                double xRatio, double cxRatio,
double yRatio, double cyRatio )
{
      CItemCtrl   item;
      CRect       cltRect;

      GetDlgItem(nID)->GetWindowRect(&item.m_wRect);
      ScreenToClient(&item.m_wRect);

      item.m_nID = nID;
      item.m_stxLeft = xl;
      item.m_stxRight = xr;
      item.m_styTop = yt;
      item.m_styBottom = yb;
      item.m_bFlickerFree = !!(bFlickerFree & 0x01);
      item.m_bInvalidate = !!(bFlickerFree & 0x02);
      item.m_xRatio = xRatio;
      item.m_cxRatio = cxRatio;
      item.m_yRatio = yRatio;
      item.m_cyRatio = cyRatio;

      GetClientRect(&cltRect);
      if (xl == CST_RELATIVE || xl == CST_ZOOM || xr == CST_RELATIVE || xr ==
CST_ZOOM)
            item.m_xRatio = (item.m_wRect.left + item.m_wRect.right) / 2.0 /
cltRect.Width();

      if (yt == CST_RELATIVE || yt == CST_ZOOM || yb == CST_RELATIVE || yb ==
CST_ZOOM)
            item.m_yRatio = (item.m_wRect.bottom + item.m_wRect.top ) / 2.0 /
cltRect.Height();

      std::vector<CItemCtrl>::iterator it;
      int  nCount;

      if((nCount = m_Items.size()) > 0) {
            for (it = m_Items.begin(); it != m_Items.end(); it++) {
                  if (it->m_nID == item.m_nID) {
                        *it = item;
                        return;
                  }
            }
      }

      m_Items.push_back(item);

}

void CResizeDlg::AllowSizing(int xst, int yst)
{
```

```
	m_xSt = xst;
	m_ySt = yst;
}

void CResizeDlg::HideSizeIcon(void)
{
	m_wndSizeIcon.ShowWindow(SW_HIDE);
}

int CResizeDlg::UpdateControlRect(UINT nID, CRect *pnr)
{
	std::vector<CItemCtrl>::iterator it;
	int  nCount;

	if ((nCount = m_Items.size()) > 0) {
		for (it = m_Items.begin(); it != m_Items.end(); it++) {
			if (it->m_nID == nID) {
				if (pnr != NULL) {
					it->m_wRect = *pnr;
				}
				else {
					GetDlgItem(nID)->GetWindowRect(&it->m_wRect);
					ScreenToClient(&it->m_wRect);
				}

				return 0;
			}
		}
	}

	return -1;
}
```

RuleDlg.cpp – Установка умов

```
//Описувач класу програми
//Підключення основних оголошень діалогового вікна
#include "stdafx.h"
#include "SIEM_App.h"
#include "RuleDlg.h"

#include "sockUtil.h"

// Ініціалізація дебаг інформації
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CRuleDlg діалог

//Опис конструктора
CRuleDlg::CRuleDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CRuleDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CRuleDlg)
  //Ініціалізація первісних значень
	m_ipsource = _T("0.0.0.0");
	m_portsource = 0;
	m_ipdestination = _T("0.0.0.0");
	m_portDestination = 0;
	m_action = _T("Tirar");
	m_protocol = _T("TCP");
	m_srcMask = _T("0.0.0.0");
	m_dstMask = _T("0.0.0.0");
	//}}AFX_DATA_INIT

	defaultAction = PF_ACTION_FORWARD;
}

//функція зчитування й установки даних вікна
void CRuleDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CRuleDlg)
	DDX_Control(pDX, IDC_COMBO3, m_actionCombo);
	DDX_Text(pDX, IDC_EDIT1, m_ipsource);
	DDV_MaxChars(pDX, m_ipsource, 15);
	DDX_Text(pDX, IDC_EDIT2, m_portsource);
	DDX_Text(pDX, IDC_EDIT3, m_ipdestination);
	DDV_MaxChars(pDX, m_ipdestination, 15);
	DDX_Text(pDX, IDC_EDIT4, m_portDestination);
	DDX_CBString(pDX, IDC_COMBO3, m_action);
	DDX_CBString(pDX, IDC_COMBO4, m_protocol);
	DDX_Text(pDX, IDC_EDIT6, m_srcMask);
	DDV_MaxChars(pDX, m_srcMask, 15);
	DDX_Text(pDX, IDC_EDIT5, m_dstMask);
	DDV_MaxChars(pDX, m_dstMask, 15);
	//}}AFX_DATA_MAP
}

//Мапінг Windows подій, перехоплення пост повідомлень
BEGIN_MESSAGE_MAP(CRuleDlg, CDialog)
	//{{AFX_MSG_MAP(CRuleDlg)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CRuleDlg заголовок повідомлень
```

```
//обробка подій (пост повідомлень) Windows
void CRuleDlg::OnOK()
{
	int result;

  //Зчитування, перевірка на коректність вводу
  //вивід помилок із описом проблем
	UpdateData(TRUE);

	result = inet_addr(m_ipsource, &srcIp);

	if(result == -1)
	{
		AfxMessageBox("Source Address isn't valid.");

		return;
	}

	if(srcIp == 0)
		srcMask = 0;
	else
	{
		result = inet_addr(m_srcMask, &srcMask);

		if(result == -1)
		{
			AfxMessageBox("Source Mask isn't valid.");

			return;
		}
	}


	result = inet_addr(m_ipdestination, &dstIp);

	if(result == -1)
	{
		AfxMessageBox("Destination Address isn't valid.");

		return;
	}

	if(dstIp == 0)
		dstMask = 0;

	else
	{
		result = inet_addr(m_dstMask, &dstMask);

		if(result == -1)
		{
			AfxMessageBox("Destination Mask isn't valid.");

			return;
		}
	}

	srcPort = m_portsource;
	dstPort = m_portDestination;

	if(m_protocol == "TCP")
		protocol = 6;

	else if(m_protocol == "UDP")
		protocol = 17;

	else if(m_protocol == "ICMP")
	{
		protocol = 1;
```

```
            if(srcPort == 0x00)
                  srcPort = 0xff;

            if(dstPort == 0x00)
                  dstPort = 0xff;

      }

      else
            protocol = 0;


      if(m_action == "")
      {
            AfxMessageBox("Select an action, please");

            return;
      }

      else
      {
            if(m_action == "Forward")
                  cAction = 0;

            else
                  cAction = 1;

      }

      //Виклик оброблювача події предка для завершення коректної реакції на
подію
      CDialog::OnOK();
}

//Ініціалізація вікна
BOOL CRuleDlg::OnInitDialog()
{
      CDialog::OnInitDialog();

  //Установка значень управління вікна
      if(defaultAction == PF_ACTION_DROP)
            m_actionCombo.AddString("Forward");

      else
            m_actionCombo.AddString("Drop");

      return TRUE;  //
}
```

sockUtil.cpp – Опис системних функцій по роботі з IP адресою перетворення форматів

```
//Підключення основних оголошень діалогового вікна
#include "stdafx.h"
#include "sockutil.h"
#include <stdlib.h>
#include <string.h>

//Опис системних функцій по роботі з IP адресами й перетворення форматів.
int inet_addr(const char *sIp, unsigned long *lIp)
{
	int octets[4];
	int i;
	const char * auxCad = sIp;
	*lIp = 0;

	for(i = 0; i < 4; i++)
	{
		octets[i] = atoi(auxCad);

		if(octets[i] < 0 || octets[i] > 255)
			return -1;

		*lIp |= (octets[i] << (i*8));

		auxCad = strchr(auxCad, '.');

		if(auxCad == NULL && i!=3)
			return -1;

		auxCad++;
	}


	return 0;
}



unsigned short htons(unsigned short port)
{
	unsigned short portRet;

	portRet = ((port << 8) | (port >> 8));

	return portRet;
}
char *IpToString(char *ip, unsigned long lIp)
{
	char octeto[4];

	ip[0] = 0;

	itoa(lIp & 0xff, octeto, 10);

	strcat(ip, octeto);
	strcat(ip, ".");


	itoa((lIp >> 8) & 0xff, octeto, 10);

	strcat(ip, octeto);
	strcat(ip, ".");
```

```
        itoa((lIp >> 16) & 0xff, octeto, 10);

        strcat(ip, octeto);
        strcat(ip, ".");

        itoa((lIp >> 24) & 0xff, octeto, 10);

        strcat(ip, octeto);


        return ip;
}
```