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
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки для захисту периметру мережі від зовнішніх атак”

Виконав здобувач вищої освіти
IV курсу, групи КБ-22-МБ
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Останній Б.В.
«____» _____ 2025 р.

Керівник проекту
доктор філософії (PhD)
_____ Дрєєва Г.М.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет
Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Галузь знань ._12 _____*"*Інформаційні технології*"*_____
Спеціальність *125 "*Кібербезпека*"*
Освітньо-професійна (освітньо-наукова) програма _*"*Кібербезпека*"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2025 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Останньому Богдану Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки для захисту периметру мережі від зовнішніх атак*

2. Керівник роботи *Дрєєва Ганна Миколаївна, доктор філософії (PhD)*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 51-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту *23.05.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки для захисту периметру мережі від зовнішніх атак*

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

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи кібербезпеки в промислову експлуатацію.
6. Висновки

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

Структурна схема системи кібербезпеки	*1 аркуш*
Функціональна схема системи кібербезпеки	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Дрєєва Г.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Останній Б.В.
(прізвище та ініціали)

АНОТАЦІЯ

Останній Б.В. Програмне забезпечення системи кібербезпеки для захисту периметру мережі від зовнішніх атак. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки для захисту периметру мережі від зовнішніх атак.

Метою розробки є програмне забезпечення системи кібербезпеки для захисту периметру мережі від зовнішніх атак.

Результат роботи – програмна реалізація системи кібербезпеки для захисту периметру мережі від зовнішніх атак.

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 з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: кібербезпека, захист периметру мережі

ABSTRACT

Ostanniy B.V. Software for a cybersecurity system to protect the network perimeter from external attacks. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed that is intended for a cybersecurity system to protect the network perimeter from external attacks.

The purpose of the development is software for a cybersecurity system to protect the network perimeter from external attacks.

The result of the work is a software implementation of a cybersecurity system to protect the network perimeter from external attacks.

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.

A convenient user interface has been developed. Instructions for working with software are provided.

The program can be used on a PC with OS Windows 10/11.

The program was developed in the Python environment.

Keywords: cybersecurity, network perimeter protection

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЗІ	–	захист інформації
ІБ	–	інформаційна безпека
ІТ	–	інформаційна технологія
КІС	–	корпоративні інформаційні системи
ЛОМ	–	локальна обчислювальна мережа
НСД	–	несанкціонований доступ
ОС	–	операційна система
СГ КІС	–	сегмент КІС
СЗІ	–	система захисту інформації
IPS	–	система запобігання вторгнень
NAC	–	Network Admission Control
NIDS	–	система виявлення мережних вторгнень

ВСТУП

Актуальність теми. Хоча поняття периметра корпоративної мережі за останні роки перетерпіло значні зміни, його захист залишається обов'язковим елементом інформаційної безпеки організації й важливої складової багаторівневої системи, що допомагає звести до мінімуму зовнішні погрози. Однак її вже недостатньо: сучасні «мережі без кордонів», хмарна модель обчислень і мобільність користувачів вимагають нових підходів.

Традиційно рішення для захисту периметра – зовнішньої границі мережі – застосовуються в організаціях при підключенні корпоративних мереж до мереж загального користування.

Вони дозволяють запобігти атакам на IT-ресурси й реалізувати безпечний доступ співробітників компаній у зовнішні мережі, а авторизованих віддалених користувачів – до корпоративних ресурсів.

Захист периметра вважається обов'язковим елементом системи забезпечення інформаційної безпеки корпоративної мережі й містить у собі шлюзи безпеки, засобу міжмережного екранування (FW), організацію віртуальних приватних мереж (VPN), системи виявлення й запобігання вторгнень (IDS/IPS). Її реалізація залишається однією з основних завдань ІБ і основою надійного функціонування критичних для компанії інформаційних систем.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки для захисту периметру мережі від зовнішніх атак.

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

– Огляд існуючих систем для захисту периметру мережі від зовнішніх атак.

– Дослідження системи кібербезпеки для захисту периметру мережі від зовнішніх атак.

– Програмна реалізація системи кібербезпеки для захисту периметру мережі від зовнішніх атак.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі для захисту периметру мережі від зовнішніх атак.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для захисту периметру мережі від зовнішніх атак, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Міжмережні екрани й граничні маршрутизатори із правильно настроєною конфігурацією – перша лінія оборони, що запобігає несанкціонований доступ у корпоративну мережу. На зміну міжмережним екранам колишнього покоління (з фільтрацією пакетів), що блокують лише мережні порти й IP- і MAC-адреси, прийшли нові системи з функціями забезпечення безпеки на рівні додатків, на якому зараз здійснюється більшість атак.

Засоби міжмережного екранування забезпечують не тільки захист від атак, але й захищене з'єднання між офісами, а також безпечний віддалений доступ співробітників до корпоративних IT-ресурсам. Їх доповнюють шлюзи безпеки, здатні підтримувати велика кількість захищених каналів зв'язку.

Ще один клас продуктів – системи виявлення/запобігання вторгнень (IDS/IPS). Вони дозволяють проводити глибокий аналіз активності в мережі на всіх рівнях моделі OSI, обновляти в реальному часі бази сигнатур атак і ознак вторгнень, забезпечувати захист від уразливостей нульового дня за допомогою адаптивних технологій перевірки.

Для організації захищених каналів зв'язку між територіально розподіленими офісами компанії звичайно застосовується технологія VPN. Такі довірені канали із шифруванням трафіку також включаються в захищений периметр мережі. Але де тепер проходить її границя?

1.2 Область застосування

З огляду на стрімке поширення мобільних пристроїв, концепції BYOD, хмарних обчислень і різних технологій для віддаленої роботи, всі частіше

доводиться чути про зникнення периметра корпоративної мережі, однак концепція його захисту не застарів, вона лише має потребу в адаптації до сучасних умов. Деякі зважаться відмовитися від міжмережних екранів і шлюзів безпеки.

Проте використання хмарних обчислень, мобільних пристроїв і віртуалізації приводить до розмивання традиційного захисту периметра – її доводиться поширювати як за межі, так і усередину мережі. Це явище одержало назву «деперіметризація»: з розширенням способів доступу до корпоративних ресурсів і додатків у мережі більше немає єдиної точки входу. До того ж нові технології й тенденції вимагають інших підходів до організації захисту корпоративної мережі.

Концепція традиційного периметра як чітко обкресленої й незмінної границі мережі усе ще застосовується тими організаціями, де з підозрою відносяться до новомодних IT-технологій – хмарам, BYOD, «Інтернету речей» і т.п. Насамперед це військові структури, деякі державні органи, а також установи, що обробляють засекречені відомості. У більше сучасних організаціях поняття традиційного периметра дійсно зникає, йому на зміну приходить «нечіткий», або «розмитий», периметр: границя мережі динамічно міняється й проходить по мобільних пристроях і хмарній інфраструктурі, де зберігаються інформаційні активи, які захищаються.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для захисту периметру мережі від зовнішніх атак, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

BestCrypt

Однією з основних особливостей даної програми є те, що вона дозволяє створювати контейнер у контейнері. Інша особливість в тому, що програма – мультиплатформена. Є версії для Windows і для Linux.

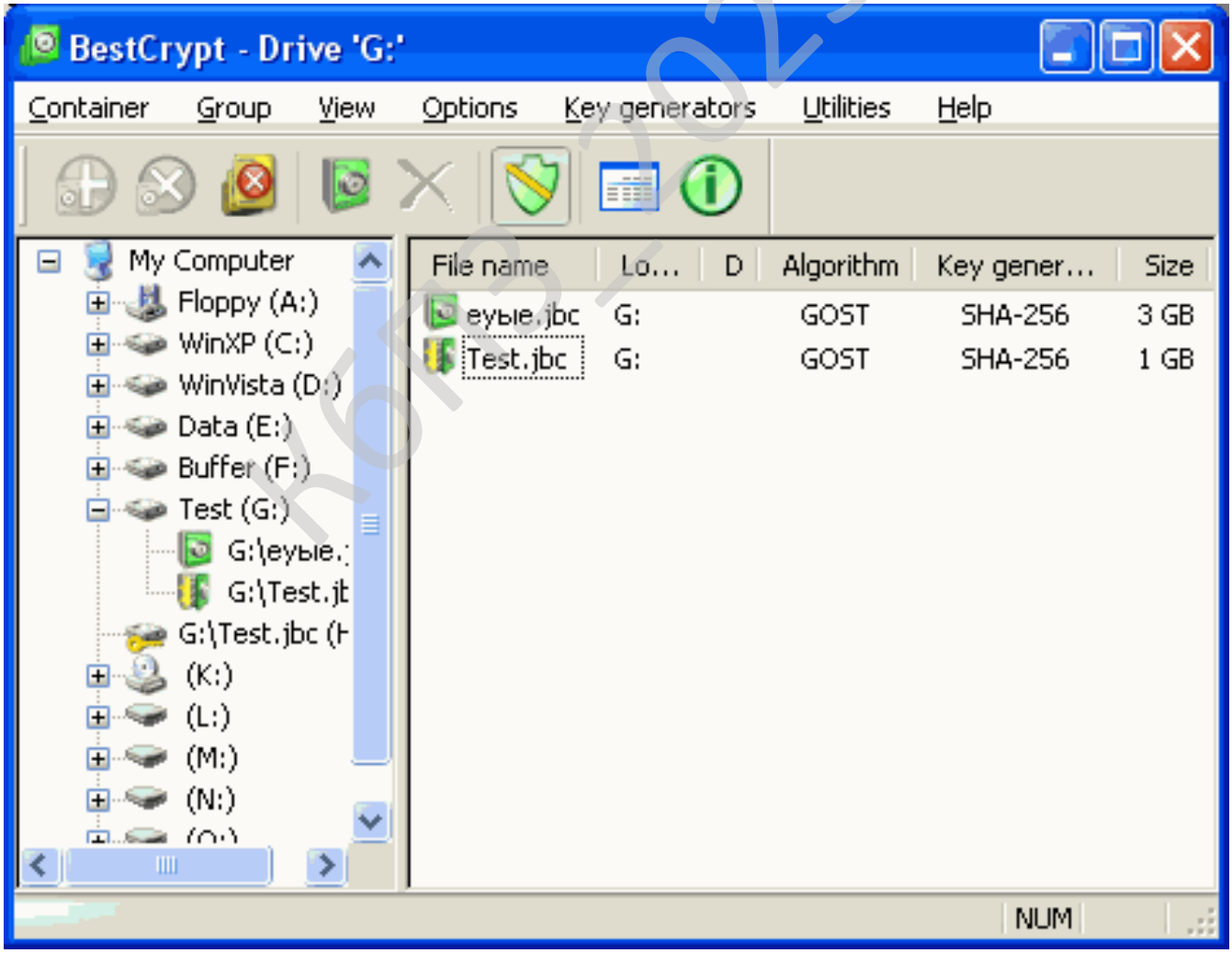

Рисунок 2.1 – BestCrypt – Інтерфейс програми

Із приводу використання програм подібного типу для збереження даних в Інтернеті ведеться багато суперечок. Хтось уважає, що програми, сертифіковані компетентними органами, обов'язково мають програмні "закладки" для доступу до інформації. Хтось уважає, що "закладки" не потрібні, тому що компетентні органи й так одержать паролі особливими методами. Ми не вдамося в ці суперечки, ми розглядаємо програму як засіб для зберігання інформації від несанкціонованого доступу.

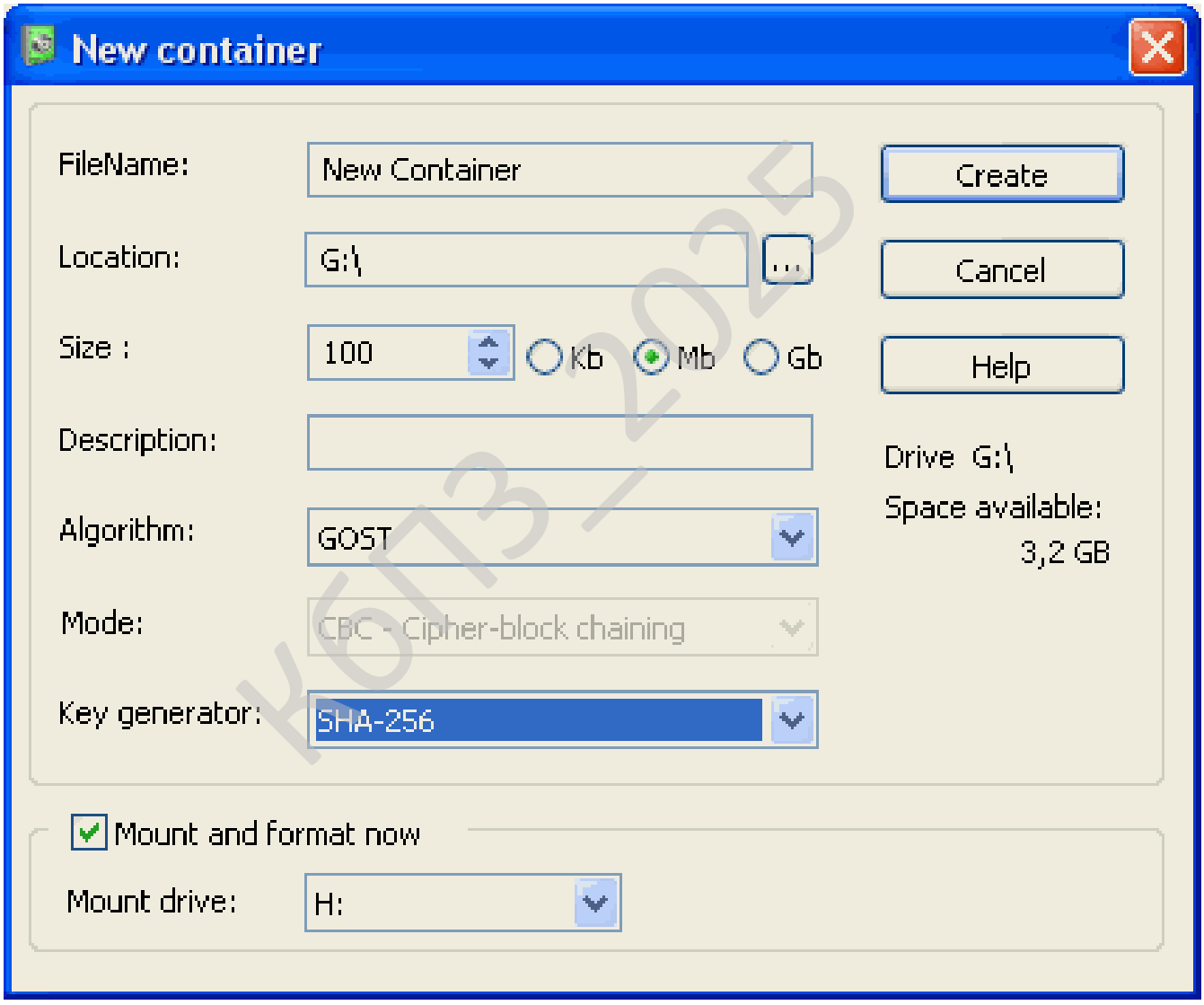

Рисунок 2.2 – Діалог створення нового контейнера

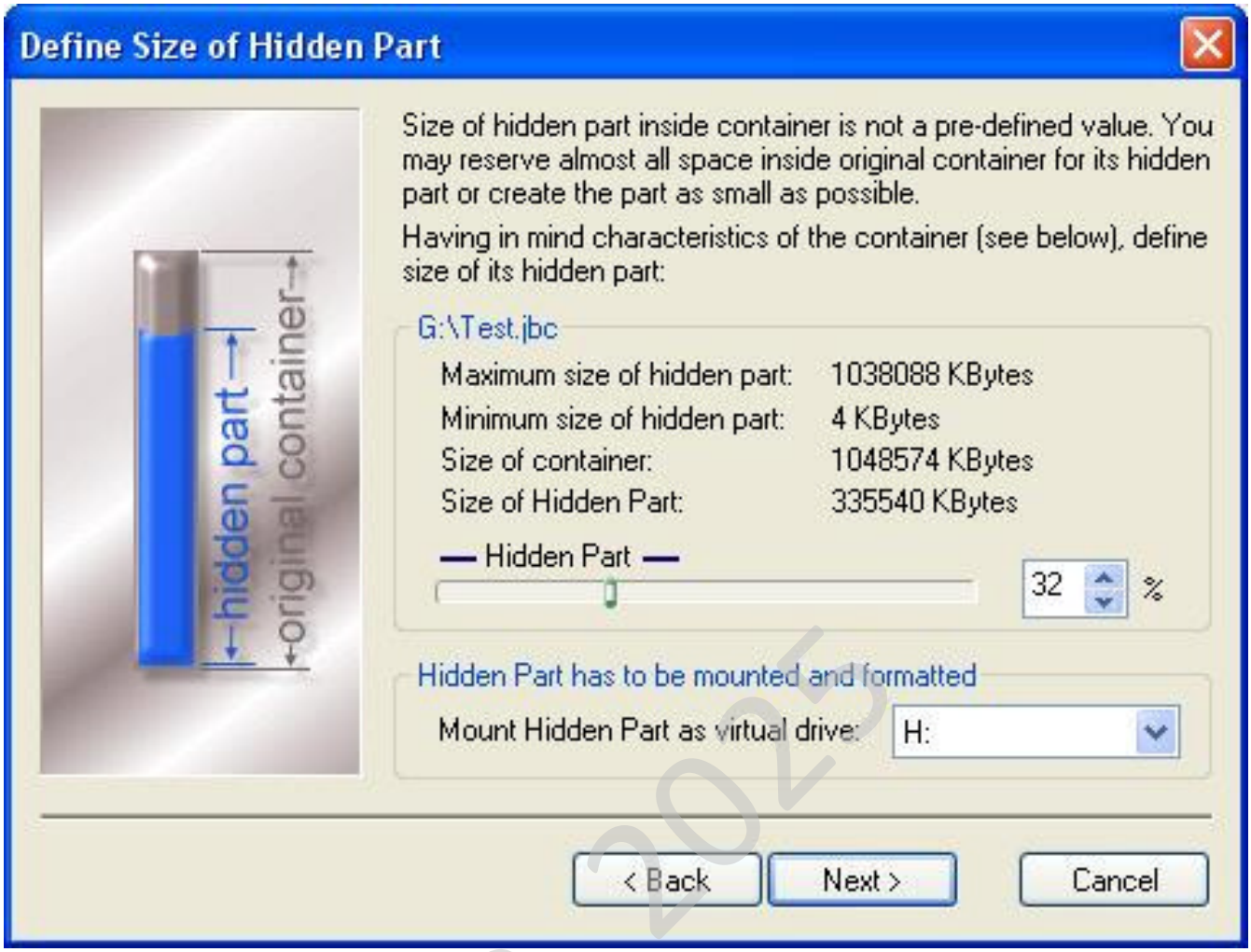

Рисунок 2.3 – Створення схованого контейнера

Для того щоб швидко при необхідності розмонтувати контейнери, краще використовувати "гарячі" клавіші. А для забудькуватих (тих, хто може відлучитися від комп'ютера, не розмонтувавши диски) радимо настроїти функцію автоматичного розмонтування після закінчення певного часу.

Як уже говорилося, створювані контейнери можна зберігати на будь-якому носії, дублювати, переміщати. Захист від випадкового видалення в цьому плані доводиться дуже до речі, тому що вона працює на всіх носіях, установлених на комп'ютері. Тобто, переміщаючи або копіюючи контейнери, ви зберігаєте за ними захист від видалення. Захист поширюється на DOS-сесію або спробу видалити контейнер, використовуючи DOS-додатки.

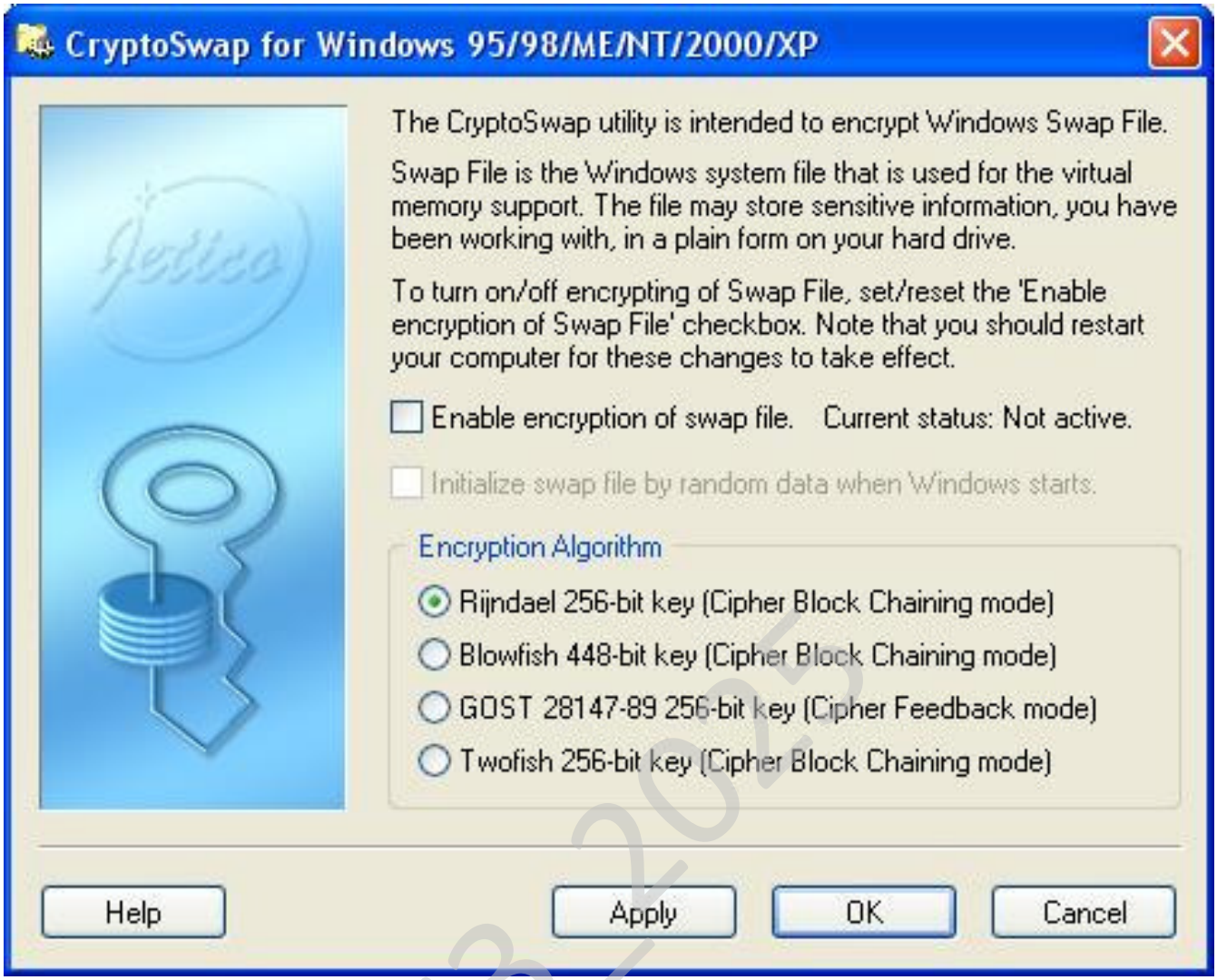

Рисунок 2.4 – Шифруємо свап-файл

За увесь час існування програми не було замічено проблем несумісності контейнерів, створених більше ранніми версіями цього програмного забезпечення. Єдине, що потрібно відзначити, – при роботі з контейнерами потрібно бути обережними в тім плані, що програма не завжди коректно розуміє ситуацію тимчасового приєднання інших дисків. Можуть збігатися літери контейнерів BestCrypt і дисків, що додаються, якщо контейнери не були змонтовані в систему автоматично. У цьому випадку система може не побачити контейнери. У цілому ж по досвіду багатьох користувачів можна говорити про

програму як про дуже надійне й практичне рішення для зберігання конфіденційної інформації.

Настав час нормалізувати конфіденційність

Видобуток даних повністю вийшов з-під контролю. Хоча смартфони досить погані, Apple і навіть Google вдалося дати нам більше контролю над тим, якими даними вони обмінюються. Але є й інші поширені пристрої, які відстежують нас, про які ви можете не знати, як-от ваш автомобіль і телевізор. Без законів, які це забороняють, технологічні компанії та брокери даних агресивно збирають вражаючі обсяги особистих даних, щоб монетизувати їх. Навіть якщо ви не думаєте, що дбаєте про власну конфіденційність (а якщо ні, будь ласка, прочитайте *Конфіденційність – це сила*), ваша конфіденційність перетинає конфіденційність тих, хто вас оточує. А порушення конфіденційності часто призводять до збоїв у безпеці.

Поки ми не встановимо норми, що захищатимуть наше право на конфіденційність, ми можемо зробити багато досить простих речей, які обмежуватимуть наш вплив. І що важливіше, нам потрібно нормалізувати конфіденційність і використання технологій збереження конфіденційності. Конфіденційність полягає не в тому, щоб мати щось приховувати, а в тому, щоб ділитися лише тим, чим ви хочете поділитися, і лише з ким ви хочете цим поділитися. Конфіденційність є основним правом людини. Вам не потрібно обґрунтовувати право, воно просто *є*. Але якщо ви не *відстоюєте* свої права, ви можете підірвати їх не лише для себе, але й для інших, які можуть залежати від них набагато більше, ніж ви.

У 2025 році прийміть рішення нормалізувати конфіденційність! Це має бути типовим, фактичним стандартом. Нижче наведено кілька порад, які допоможуть зменшити ваш цифровий слід і захистити ваші дані. Подивіться, скільки ви можете перевірити цього року! (А коли ви закінчите, допоможіть своїм близьким зробити те саме.)

Мінімізуйте свій вплив

Одним із головних проектів, які я взявся у 2024 році, було знайти та переглянути якомога більше моїх загальнодоступних даних. Навіть я був шокований тим, скільки ще було там. Зараз, як власнику бізнесу і публічній особі, дуже важко бути анонімним – ці дві речі прямо суперечать. Але навіть поза межами моєї професійної інформації я все одно знайшов масу особистих даних, які не мали ніякого значення. Проблема в тому, що велика частина наших особистих даних розміщена на державних веб-сайтах: майнові та податкові записи, реєстрація виборців, цивільні та кримінальні позови, записи про шлюби та розлучення тощо. І існують сотні, якщо не тисячі брокерів даних і фірм з «управління ризиками», які шукають, збирають, порівнюють, співвідносять і пакують цю інформацію для продажу будь-кому, хто готовий заплатити. І, на жаль, ви можете отримати багато чого з цього, навіть не заплативши, якщо добре пошукати. Ось кілька способів, якими ви можете (і повинні) зменшити доступ до інформації:

– Знайдіть і видаліть свої публічні дані. Єдині дані, які не можна вкрасти чи зловживати, це дані, яких не існує. Вам слід спробувати зменшити обсяг загальнодоступних даних про вас. Хоча це можна зробити самостійно, легше заплатити за це комусь іншому. Подивіться мою серію статей, які пояснюють, де знайти вашу інформацію та як її зменшити або видалити.

– Видалити непотрібні облікові записи. Проведіть інвентаризацію своїх онлайн-акаунтів і видаліть усе, що вам більше не потрібно. Якщо можливо, попросіть їх також видалити ваші дані. Єдиним можливим винятком є облікові записи соціальних мереж і електронної пошти: ваш ідентифікатор користувача може бути перероблено (надано комусь іншому), після чого вони зможуть надсилати й отримувати електронні листи за вашим старим обліковим записом. Перегляньте цю статтю для отримання додаткової інформації.

– Заморозити ваш кредит. Насправді немає причин не заморозити ваш рахунок у великих кредитних бюро. Це безкоштовно та легко заморожувати та

розморожувати скільки завгодно часто, і це не впливає на ваш кредитний рейтинг. Це один із найкращих способів запобігти крадіжці особистих даних.

– Дегуглите своє життя. Google є рекламною компанією – 80-90% їх доходу припадає на рекламу. Окрім усіх продуктів із «Google» у назві, Google володіє Waze, Fitbit, YouTube, Android, браузер Chrome та багато іншого. Є альтернативи, які поважають конфіденційність – подивіться, скільки продуктів Google ви можете замінити.

– Закрийте Chrome. Якщо ви збираєтеся вибрати лише один продукт Google для заміни, зробіть це Chrome. Chrome від Google є найпопулярнішим веб-браузером на планеті. Але використання Chrome дає Google повний доступ до всього, що ви робите в Інтернеті. У Chrome є налаштування конфіденційності, але їх недостатньо – нещодавно в Chrome з'явилися сторонні блокувальники реклами. Є просте рішення: відмовтеся від Chrome і використовуйте веб-переглядач, який поважає конфіденційність.

– Блокувати рекламу (і відстеження). Я розумію, що реклама (і монетизація ваших особистих даних) – це те, що платить за багато «безкоштовних» продуктів. Але рекламні компанії не просто показують вам рекламу, вони збирають ваші дані, щоб показувати вам *націлену* рекламу. Ви повинні заблокувати це відстеження, де це можливо. А видалення цієї надокучливої реклами зробить роботу з Інтернетом *набагато* приємнішою – ви будете дивуватися, чому не зробили цього раніше.

– Вимкнути обмін даними AI. Кожен додає певну форму «ШІ» (штучного інтелекту) до своїх продуктів. Але в гонці за найпотужнішими та всезнаючими функціями штучного інтелекту компанії, які створюють ці продукти, часто використовують ваші запити, електронні листи, повідомлення та документи для навчання своїх моделей штучного інтелекту наступного покоління. Цю «функцію» часто ввімкнено за замовчуванням, і налаштування для її вимкнення рідко є очевидними – і часто навмисно приховуються. Деякі приклади можна побачити тут. Я збираюся перевірити, чи зможу я незабаром зібрати статтю про вимкнення збору даних ШІ, тому слідкуйте за оновленнями.

Нормалізація шифрування

З самого початку Інтернету уряди (зокрема правоохоронні та розвідувальні служби) намагалися обмежити використання технології шифрування. У 90-х ми назвали це Crypto Wars, що призвело до створення новаторських технологій, таких як Pretty Good Privacy (PGP). Сьогодні ці агентства все ще скаржаться на те, що вони замовчують, і вимагають «відповідально керованого шифрування». Але немає чорного ходу, який пропускає лише хороших хлопців, як нам щойно дуже болісно нагадали.

Ми всі потребуємо та заслуговуємо на надійне шифрування та справді приватне спілкування, яке воно забезпечує. Це має бути нормою, а не винятком. Ви не повинні робити нічого злочинного чи негідного. Кожен має використовувати безпечні та приватні засоби зв'язку щодня та постійно. Просто немає причин не робити цього. Запропонуйте своїм друзям і родині зробити це також:

– Використовуйте сигнал. Програма Signal досі є моєю основною рекомендацією для обміну приватними повідомленнями. Є кілька інших варіантів, які пропонують справжнє наскрізне шифрування (E2EE), і ви можете обговорювати, чи той чи інший інструмент кращий для екстремальних ситуацій. Але Signal – це проста відповідь для переважної більшості звичайних людей. Єдиним мінусом є те, що вам доведеться переконувати інших приєднатися до вас. Одна перевага: це акуратно вирішує проблему синьої чи зеленої бульбашки. Signal працює на всіх платформах: iOS, Android, macOS, Windows і Linux.

– Використовуйте Proton все. Proton Mail була фантастичною службою, коли її запустили 10 років тому. Він пропонував справді приватну електронну пошту. Але за останні роки Proton додав багато більш безпечних і приватних послуг: VPN, Drive (безпечне хмарне сховище), Pass (менеджер паролів), Calendar, Docs і навіть гаманець Bitcoin. Знову ж таки, існують інші служби, але мати їх усі під однією парасолькою просто дуже зручно. Щоб отримати максимальну віддачу від цих послуг, вам також знадобляться ваші друзі та

родина, щоб отримати облікові записи Proton, але для початку вони мають дуже зручний безкоштовний рівень. Вони також пропонують сімейні облікові записи.

– Шифруйте свої диски. На щастя, більшість смартфонів сьогодні мають зашифровані диски за замовчуванням, якщо ви налаштували їх на блокування (що потрібно). Ви можете легко зашифрувати комп'ютери Mac за допомогою FileVault. На жаль, Windows Home не постачається з інструментом Microsoft для шифрування дисків BitLocker – вам знадобиться Pro або Enterprise. Але ви можете використовувати VeraCrypt, який є безкоштовним.

– Використовуйте Cryptomator. Популярні хмарні сховища, такі як Microsoft OneDrive, Google Drive і Dropbox, технічно зашифровані, але ключі зберігаються у постачальників послуг. Якщо ви хочете й надалі ними користуватися, подумайте про те, щоб помістити ваші найбільш конфіденційні документи в спеціально зашифровану папку «сховище» за допомогою такого інструменту, як Cryptomator (VeraCrypt також може це зробити).

– Використовуйте VPN (де це має сенс). Віртуальні приватні мережі не такі приватні, як ви могли б подумати, і вони точно не є срібною кулею для конфіденційності, як багато хто заявляє, що є. Особисто я використовую VPN лише тоді, коли моя точка доступу мобільного телефону не працює. Якщо ви все-таки вирішите використовувати VPN, використовуйте таку, яка дійсно поважає вашу конфіденційність.

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

Python – це потужна мова програмування, яка проста у вивченні. Він має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис і динамічна типізація Python разом з його інтерпретованим характером роблять його ідеальною мовою для створення сценаріїв і швидкої розробки додатків у багатьох

сферах на більшості платформ. Інтерпретатор Python і обширна стандартна бібліотека доступні у вихідному або двійковому вигляді для всіх основних платформ на веб-сайті Python https://www.python.org/ і можуть вільно поширюватися. Цей же сайт також містить дистрибутиви та вказівники на багато безкоштовних сторонніх модулів Python, програм і інструментів, а також додаткову документацію.

Інтерпретатор Python легко розширюється за допомогою нових функцій і типів даних, реалізованих у C або C++ (або інших мовах, які можна викликати з C). Python також підходить як мова розширення для налаштовуваних програм.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки для захисту периметру мережі від зовнішніх атак.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) 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 кібербезпеки 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 кібербезпеки 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

Концепція захисту периметра поки не застаріла, але все до цього йде. Ландшафт IT-інфраструктури стрімко міняється, і фахівцям з IT і ІБ необхідно адаптуватися до цих змін. Треба чітко розуміти, у кого є права віддаленого доступу до ресурсів мережі, кому й що можна обробляти на мобільних пристроях, які дані зберігаються в хмарах, і, виходячи із цього, оцінювати можливі ризики й ухвалювати рішення щодо керування ними. Одні підуть по шляху заборони, а інші організують моніторинг або запропонують користувачам зручні й захищені сервіси. У цілому використання зовнішніх сервісів і персональних пристроїв для обробки корпоративної інформації усе ще не дуже поширене в Україні.

Однак, за даними закордонної статистики, більше половини ділових комунікацій і транзакцій сьогодні вже здійснюється поза корпоративною мережею. Малі компанії можуть обійтися й без традиційної IT-інфраструктури, використовуючи хмарні сервіси (Iaa) і клієнтські пристрої, а середні й великі організації вибирають для себе аутсорсинг IT, BYOD, хмарні додатки й різні способи доставки додатків і даних. Забезпечити захист корпоративних даних стає усе складніше. Зміни, що відбуваються зараз у даній області, напевно, самі значні за всю історію інформаційної безпеки.

З розвитком технологій мобільних пристроїв ситуація кардинально змінилася. Співробітники застосовують смартфони, планшети, торговельні термінали й інші пристрої, які використовують різні способи підключення до корпоративної мережі. Тому необхідно вдосконалювати системи безпеки з урахуванням нових потреб бізнесу – передбачити виникнення нових ризиків і впровадити додаткові технічні засоби оборони. Для захисту мобільних пристроїв

і керування ними потрібно застосовувати новий клас рішень, які дозволяють забезпечити безпечне зберігання даних на цих пристроях і контроль доступу до них.

Традиційна модель добре працює, коли 90% користувачів перебувають в офісі й в 90% випадків доступ здійснюється до додатків, установленим на серверах, які розташовані на власній площадці компанії. Хмари й мобільність зовсім міняють картину. Мобільні користувачі виходять в Інтернет, минаючи корпоративні шлюзи й міжмережні екрани, а застосування зовнішніх систем електронної пошти, мобільних додатків і соціальних мереж збільшує ризик витоку важливих даних. Очевидно, потрібний новий підхід, що забезпечує безпечну роботу, де б користувач не перебував і який би пристрій не застосовував.

Розвиток інформаційних технологій, безумовно, вимагає застосування нових підходів до забезпечення безпеки. Поняття периметра мережі в цей час істотно відрізняється від реалій чотирьох-п'ятирічної давнини. Мобільність співробітників стала стандартом де-факто, що вже визнано регуляторами й відбите, наприклад, у відповідних нормативно-методичних документах України у вигляді вимог до захисту мобільних пристроїв.

Традиційні рішення в області мережної безпеки залишаються актуальними – це базовий набір сервісів для захисту інфраструктури. Крім того, у деяких випадках обов'язковість їхньої наявності закріплена законодавчо. Зокрема, це стосується питань захисту персональних даних. Але у зв'язку зі збільшенням числа мобільних користувачів концепція захисту периметра перетерплює деякі зміни. Всі частіше сучасним організаціям на додаток до систем міжмережного екранування, системам запобігання вторгнень і інших засобів ІБ потрібні рішення класу BYOD.

На зміну фізичному мережному периметру в класичному розумінні цього терміна прийшло поняття «віртуальний периметр», що охоплює всю інформаційну екосистему компанії: ЦОД, мобільні пристрої, ноутбуки, канали

передачі даних і т.д. Тому не можна затверджувати, що периметра більше немає, скоріше відбулося його значне розширення. Разом з тим розмивання фізичних кордонів істотно підвищило ризики, пов'язані із захистом конфіденційних даних, і вивело питання безпеки на новий щабель. Зрослі вимоги до можливостей централізованого керування компонентами системи сприяють появі на ринку нових рішень.

3.2 Розробка структурної схеми

Концепція захисту периметра видозмінюється відповідно до нових технологій. Нова модель доступу до корпоративної інформації породжує нові погрози, а виходить, висуває додаткові вимоги до засобів захисту, що стає більше складної й гранульованою. Один із прикладів – технології захищених контейнерів у мобільних пристроях. Як повинна будуватися система мережної безпеки? По-перше, потрібно забезпечити захист усе ще актуального «класичного» периметра мережі. По-друге, захистити канали передачі інформації за допомогою технологій VPN для мобільних пристроїв. У віртуальних середовищах необхідне застосування віртуальних шлюзів безпеки з функціоналом UTM.

У нових умовах акценти міняються: не можна фокусуватися винятково на периметрі, адже погроза може з'явитися звідки завгодно – проникнути в мережу з Інтернету, через флешку, ноутбук гостьового користувача або аудитора, через несанкціоновано підключений LTE-модем або точку бездротового доступу. Таким чином, треба контролювати все, що відбувається в мережі – зовні (у хмарі або на мобільних пристроях), на її границі, у ЦОД, у внутрішній локальній мережі. Тільки при такій умові можна розраховувати на ефективний захист від цілеспрямованих і схованих атак, що превалюють в останні два-три роки.

В Cisco довгі роки захист периметра будувався на базі звичайних маршрутизаторів, що виконують функції міжмережного екрана, у той час як багато виробників заявляли про обов'язковість установки окремого

спеціалізованого пристрою. Сьогодні погрози й середа, яка захищається, міняються так динамічно, що неможливо сформувати фіксований список обов'язкових захисних технологій і засобів. Все залежить від ризиків, які приймає на себе замовник.

Для побудови ефективної системи безпеки необхідно визначити, яка інформація являє цінність для організації, які сервіси й системи повинні бути доступні кінцевим користувачам, які методи доступу вони воліють. Наступним кроком повинні стати оцінка існуючого стану ІБ і виявлення можливих ризиків. І вже виходячи із цього необхідно розробити концепцію й плани розвитку ІТ і системи ІБ. «Гарним тоном» для середніх і великих організацій є використання систем моніторингу й контролю вхідного/вихідного трафіку на найвищих рівнях моделі OSI (розуміння змісту інформаційних повідомлень) і систем кореляції подій безпеки.

Як повинне виглядати цілісне рішення для захисту даних і додатків і забезпечення безпечної роботи співробітників, де б вони не перебували? Насамперед воно повинне надавати захищене середовище для роботи з корпоративними даними й додатками й бути максимально зручним для користувача. Для цього, як правило на кінцевих пристроях, установлюються спеціалізовані агентські додатки, взаємодіючі з корпоративними системами захисту. До обов'язкових засобів відносяться в першу чергу SSL VPN, рішення для захисту віртуальних середовищ, системи керування мобільними пристроями, а також інструменти для збору, аналізу й кореляції подій безпеки, захисту від спрямованих атак (APT) і т.д. Безумовно, основними блоками будуть системи керування мобільними пристроями (Mobile Device Management, MDM), додатками (Mobile Application Management, MAM) і даними (Mobile Information Management, MIM). Сучасні рішення консолідують у собі весь цей функціонал.

Проникнення, що розширюється, у корпоративне середовище мобільних пристроїв і хмарних технологій не може не впливати на принципи забезпечення безпеки інформації, що всі частіше виходить за межі периметра, що захищається.

Це підвищує значимість рішень класу MDM для забезпечення захисту інформації при роботі з мобільних пристроїв. При цьому актуальність «класичних» механізмів захисту периметра (FW, VPN і ін.) не знижується. Більше того, розвиток мережних технологій змушує динамічно розвивати їх. Вирішувати питання забезпечення безпеки інформації необхідно системно й комплексно. Важливу роль у цьому грають надійні механізми захищеного доступу, у тому числі автентифікація й захист переданих даних. Не менше значення має наявність єдиного центра керування доступом і розмежуванням прав користувачів. Наприклад, смарт-карти й токени, можуть служити персональним засобом автентифікації й електронного підпису для організації захищеного і юридично значимого документообігу. Використання смарт-карт як засіб автентифікації при роботі з інформаційними системами, Web-порталами й хмарними сервісами можливо при доступі до них не тільки з персональних робочих станцій, але й з мобільних пристроїв. Такий підхід дозволяє уніфікувати засобу автентифікації в організації, починаючи з автентифікації в операційних системах і закінчуючи системами контролю доступу в приміщення.

При побудові системи мережної безпеки важливо враховувати сучасні погрози й особливо спрямовані й DDoS-атаки. Для цього використовується цілий комплекс різних пристроїв. Компанії рівня SMB виявляють цікавість до уніфікованих систем UTM. Великі замовники можуть дозволити собі спеціалізовані, багатофункціональні й складні рішення світових лідерів. Поза залежністю від масштабу компаній все більшого значення набувають системи контентної фільтрації (URL-запитів і вхідного трафіку), як і раніше важливий захист від спама. Для захисту Web-додатків стають обов'язковими міжмережні екрани прикладного рівня (Web Application Firewall)».

Основна тенденція – зсув акценту зі звичайної фільтрації на механізми перевірки вмісту. У результаті одержують поширення міжмережні екрани з функцією контролю додатків (Application Control). Чому це відбувається? У корпоративному середовищі з'являється усе більше різних пристроїв, сервісів і

додатків – периметр «розбухає», і на перший план виходить керування єдиними правилами. У результаті починають застосовуватися шлюзи, які в автоматичному режимі дозволяють призначати й контролювати правила залежно від критичності ризиків ІБ. До складу цілісного рішення, що захищає мережу, дані, додатки, кінцеві робочі місця й іншу інфраструктуру, входять більше 20 систем, що працюють у зв'язуванні й враховуючих тенденціях в області ІБ.

В ідеалі комплексне рішення для захисту даних, додатків і користувачів повинне бути рішенням від одного вендора, що вже зарекомендовали себе на ринку засобів захисту. Такий вендор постарається швидко реагувати на зміни в сфері захисту інформації, щоб надавати новий функціонал у найкоротший термін. Компоненти системи повинні максимально коректно й зручно інтегруватися між собою й управлятися централізовано. Особливо корисна єдина консоль керування, оскільки вона дозволяє описувати правила за допомогою єдиних об'єктів мереж, хостів і користувачів. При наявності цілісного рішення адміністраторам буде набагато простіше збирати дані про події в корпоративній мережі й швидше реагувати на аномалії або атаки. А от різнорідні продукти можуть конфліктувати між собою.

Міжмережні екрани NGFW дозволяють задавати правила доступу користувачів і відслідковувати їхньої операції. Крім того, корпоративну мережу можна розділити на сегменти з різним рівнем безпеки й дозволеним доступом для мобільних або віддалених користувачів. Мобільні пристрої теж повинні бути захищеними. Для цього застосовується шифрування (наприклад, шифруються окремі розділи диска ноутбука), засобу віддаленого стирання даних (на випадок втрати або крадіжки пристрою) і т.д. Двофакторна автентифікація й шифрування трафіку в VPN допоможуть створити безпечний тунель для зовнішнього доступу й забезпечити захист мобільних і віддалених користувачів, що перебувають поза офісом.

Мобільні пристрої більше уразливі, чим сервери й настільні ПК. Крім того, що пристрій може бути украдене або загублено, на нього легко встановити

недовірені додатки. Ця область в останні роки активно розвивається: розроблювачі пропонують різноманітні рішення – від систем керування мобільними пристроями (Mobile Device Management, MDM), віртуальних машин і «контейнерів» для безпечного виконання додатків до підпису додатків і SIM-карт із убудованим електронним підписом. MDM підвищують безпека доступу мобільних користувачів до мережі – зокрема, дозволяють віддалено перевіряти смартфони, планшети й ноутбуки на відповідність корпоративній політиці безпеки, наприклад на наявність оновленої версії антивірусу.

Підходити до завданню треба системно. Дані повинні бути захищені при переміщенні, зберіганні й використанні, для чого необхідно реалізувати строге розмежування прав доступу, захист від шкідливого ПЗ, регулярне відновлення ПЗ, резервне копіювання й багато чого іншого.

Цілісне рішення повинне реалізувати чотири основних компоненти: захист робочого місця співробітника (MDM для мобільних пристроїв, VPN-клієнт, антивірус або рішення класу Endpoint Protection); захист «серверної» частини, що включає в себе засобу захисту периметра (МСЕ мережного й прикладного рівнів, шлюз VPN, системи запобігання вторгнення, контролю доступу, контентної фільтрації, захисту баз даних, моніторингу аномальної/підозрілої активності, відбиття DDoS-атак, захисту від шкідливого коду й т. п.); захист каналів передачі даних, включаючи різні технології VPN, у тому числі на базі сертифікованих засобів шифрування; засобу моніторингу й керування системою захисту, що пропонують, крім настроювання й конфігурування перерахованих вище систем, функції керування інцидентами, аналізу захищеності, керування відновленнями ПЗ й багато чого іншого.

Досвід показує, що однакові IT-інфраструктури зустрічаються дуже рідко, та й відношення до рівня критичності захищаємих даних різне. У таких умовах будуть розрізнятися як набори підсистем мережної безпеки, так і функціональність їхніх компонентів. Наприклад, якщо говорити про віддалений (у тому числі мобільний) доступ, то принципи захисти залишаються колишніми –

канали зв'язку захищаються за допомогою VPN. А при наявності технологій віртуалізації можна використовувати засоби мережного захисту, призначені для роботи в середовищі гіпервізорів.

У кожному разі вибір необхідного функціонала системи захисту залежить від профілю компанії. Типове рішення придумати складно. Для початку потрібно вивчити актуальні технології, рішення, погрози, стан поточної інфраструктури і її системи захисту – і тільки після цього визначати необхідні зміни в мережі, які пристрої вибрати для шлюзів безпеки і який функціонал на них активувати. Компанія не буде захищена належним чином, якщо використовувати тільки міжмережний екран, установлений на периметрі. Необхідно подбати й про правильне сегментування мережі, про розподіл по цих сегментах наявних ресурсів і про функціонала системи захисту для кожного з них. Шлюз для віддаленого доступу користувачів повинен передбачати різні режими автентифікації й можливість гнучкого розмежування доступу, а крім того, забезпечувати захист публікуємих ресурсів від зовнішніх погроз і атак.

На мобільних пристроях користувачів повинне бути встановлене ПЗ, що забезпечує захищена взаємодія з офісом і можливість безпечного зберігання корпоративних даних. У випадку використання смартфонів у робочих цілях необхідно, щоб застосовувані рішення дозволяли реалізувати концепцію BYOD максимально гнучко й надійно (див. рисунок 3.1). Багато організацій зупиняють свій вибір на захищеному контейнері для корпоративної пошти й конфіденційних файлів. Для компаній з банківського, телекомунікаційного й державного секторів гострим питанням залишається протидія атакам DDoS. А для віртуалізованих серверів по суті діють того ж правила захисту, що й для фізичних.

Рисунок 3.1 – Структурна схема системи

Безпека й SDN

Якщо говорити про традиційний периметр, то в програмувальних мережах засобу його захисти стають по суті марними, тому що в SDN маршрути передачі трафіку будуються незважаючи на особливості фізичної топології мережі. Разом з тим SDN цілком укладається в концепцію розмитого периметра, а в ряді випадків побудова системи захисту полегшується, оскільки в корпоративній або операторській мережі можна створити кілька центрів захисту й направляти туди трафік з будь-якої точки інфраструктури. Правда, для цього потрібні спеціалізовані рішення безпеки, що враховують специфіку SDN.

Почасти SDN з її централізованими контролерами дає відповідь на важливе питання: де саме в мережі повинна здійснюватися перевірка дотримання правил безпеки? Політикові безпеки можна більш точно співвіднести з ризиками й категоріями даних – зокрема, визначити взаємозв'язок між ресурсами або класами ресурсів, що бідують в особливому контролі ІБ. Нарешті, відповідно до підходу «безпека – це процес», гнучкість і адаптуємість програмно конфігуруємих мереж сприяють створенню постійно обновлюваної системи мережної безпеки, що відповідає новим вимогам і погрозам.

Програмно конфігуємі мережі вплинуть на захист периметра так само, як у свій час вплинула на неї віртуалізація серверів і робочих станцій, переконаний Михайло Башликів. З'являться нові погрози, пов'язані з тим, що всі потоки даних будуть оброблятися на базі одного гіпервізора, а всі дані стануть зберігатися на одному пристрої з функціями віртуалізаціи. Ці ризики прийдеться враховувати.

Розвиток SDN напевно приведе до поліпшення продуктів і сервісів систем мережного захисту: їхня масштабованість і гнучкість підвищаться, особливо при використанні в складі керованих послуг класу Managed Perimeter Security. З ростом популярності хмарних середовищ збільшується попит на дані послуги.

Ми позитивно оцінюємо Managed Perimeter Security і розраховуємо на подальший розвиток цих послуг, тому що вони дозволяють більш ефективно й з меншими витратами реалізувати захист периметра. У своєму віртуальному ЦОД ми розгорнули подібне рішення: створили центральний вузол ІБ, що надає замовникам можливість користуватися базовими сервісами за схемою SaaS.

Сервіси Managed Perimeter Security не швидко стануть популярними. Варто розрізняти два типи політик безпеки. До першого відносяться політики FW, NAT, VPN і організації мобільного доступу, тобто ті, у які досить часто доводиться вносити зміни, наприклад для розмежування доступу, тому замовник повинен сам створювати й контролювати їх. До другого типу відносяться політики IPS, антивірусного захисту, захисту від ботів, атак нульового дня – вони міняються не так часто й можуть бути представлені у вигляді якихось профілів,

але у випадку масштабної атаки вимагають дуже швидкої експертної реакції. Такого роду сервіси краще віддати на аутсорсинг – експерти допоможуть настроїти підходящий для організації профіль і швидко внесуть зміни при виникненні нової атаки або погрози.

В Україні послуги Managed Perimeter Security поки не затребувані через відсутність адекватної моделі надання аутсорсингових послуг безпеки. У нас немає компаній, готових гарантувати захист і взяти всі ризики на себе, законодавство не передбачає передачу функцій захисту в треті руки, немає надійних каналів зв'язку для забезпечення безперебійної роботи віддалених засобів захисту інформації або засобів керування ними. У таких умовах говорити про їхні перспективи передчасно.

Успішність реалізації концепції «мережі без кордонів» багато в чому залежить від розуміння того, кому необхідно надавати доступ, з яких пристроїв і до яких додатків. Практика показує, що сьогодні найбільш затребувано кілька сценаріїв. Захист може будуватися за допомогою різних підсистем для керування мобільними пристроями (MDM), контролю доступу до мережі (NAC), посиленої автентифікації й захисту каналів зв'язку (VPN). Вибираючи рішення, потрібно брати до уваги особливості інфраструктури й загальну спрямованість IT-політики компанії, а при розгляді варіантів побудови захисту – орієнтуватися на актуальний сценарій: наприклад, організацію віддаленої роботи з Web-додатками через шлюз SSL VPN, централізоване керування правилами доступу до корпоративної мережі з використанням NAC або уніфікацію доступу за рахунок віртуалізації робітників місць (VDI). Але це далеко не всі рішення: не варто забувати про такі компоненти системи захисту, як антивірус, FW, IPS, WAF, SIEM, DLP, Web- і поштові шлюзи, DAM, контроль привілейованих користувачів, і багато про що іншому.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Для подолання труднощів у слабоформалізованих ситуаціях більше високий якісний рівень захисту периметру мережі від зовнішніх атак припускає забезпечення необхідної й достатньої інтелектуальної підтримки. Запропонована в роботі функціональна схема системи інтелектуальної підтримки (СІП) захисту периметру мережі від зовнішніх атак наведена на рисунку 3.2.

У системі інтелектуальної підтримки захисту периметру мережі від зовнішніх атак пропонується використовувати інтелектуальні технології:

– механізм нечіткого логічного виводу для чисельної оцінки ймовірності атаки;

– організоване впорядкування інформації про події в базі знань;

– моделі протидії погрозам;

– прийняття рішень на вибір раціонального варіанта реагування на події безпеки.

Через необхідність максимальної структуризації розроблювальної системи й рішень, пропонується трьохрубіжна модель захисту, що щонайкраще задовольняє всієї сукупності умов її розробки, експлуатації й удосконалення. Трьохрубіжна модель захисту – неформалізований опис комплексу програмно-апаратних засобів захисту, що є основою для розробки системи захисту:

– перший рубіж – периметр об'єкта захисту – набір функціональних підсистем, що включають засоби захисту від зовнішніх вторгнень зловмисника й потенційно можливих погроз віддаленого користувача;

– другий рубіж – набір засобів захисту мережного сегмента від віддалених і локальних мережних вторгнень;

– третій рубіж містить у собі набір засобів захисту окремого персонального комп'ютера або сервера.

У процесі організаційно-технічного управління, планування ЗІ як функція управління являє собою процес послідовного зняття невизначеності щодо структури й состава засобів захисту в СЗІ. Процес планування $Р_{пл}$ раціональних наборів ЗЗ характеризується за допомогою вираження:

$$P_{\text{пл}} = \Phi \to S_r,$$

де:

– Ф – множина функціональних підсистем для рубежу захисту;

– S_r – обраний набір засобів захисту.

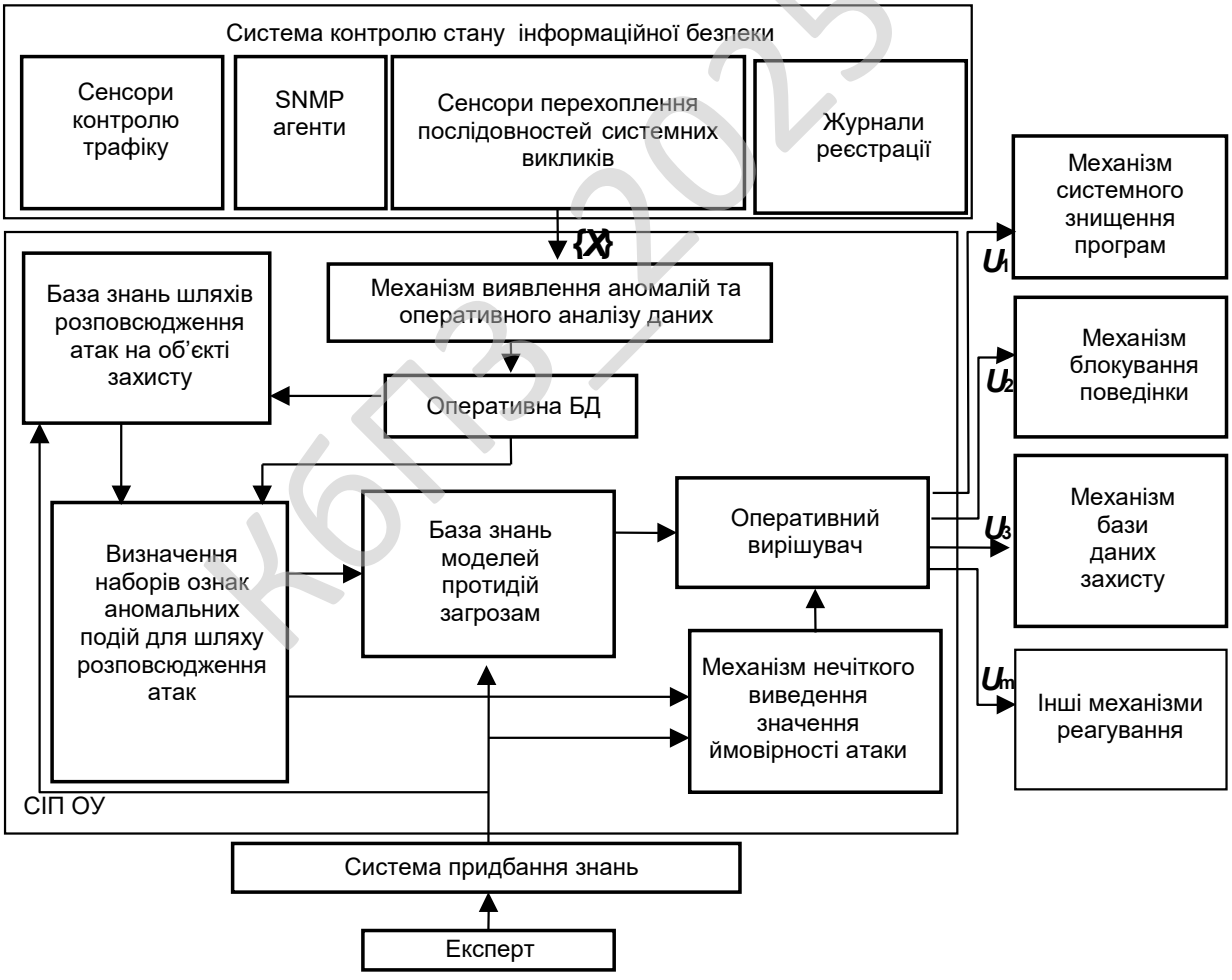

Рисунок 3.2 – Функціональна схема системи

На першому етапі задається множина функціональних підсистем для рубежів захисту, результатом планування є командна інформація, що містить конкретні дані по розподіляються ресурсам, що, що направляється на досягнення цільового стану СЗІ.

Процес ухвалення рішення про вибір раціонального варіанта набору ЗЗ для рубежу захисту – це функція перетворення змісту інформації про вимоги, запропонованих до засобів захисту, що входить у набір, про характеристики засобів захисту, у підмножину найкращих варіантів набору $S' \subseteq S$. Множина варіантів набору:

$$S = \{S_1, \ldots, S_r, \ldots, S_R\},$$

де R – число варіантів альтернативних наборів, з яких здійснюється вибір.

Для вибору раціонального варіанта набору засобів захисту використовується цільова функція J:

$$S_r = J(S).$$

Сукупність відомостей, що дозволяють зіставляти варіанти наборів, це характеристики засобів захисту функціональних підсистем для рубежу – множина W, що включає в себе дві підмножини:

$$W_{\text{зщ}_l} \subset W_l \text{ і } W_{\text{и}_l} \subset W_l,$$

де:

– $W_{\text{зщ}_l}$ – показник засобів захисту «захищеність інформації»;

– $W_{\text{и}_l}$ – показник засобів захисту «витрати» для l-ої функціональної підсистеми.

На основі морфологічного підходу модель прийняття рішень на вибір раціонального варіанта набору може бути представлена у вигляді кортежу:

$$\text{ПР: } \langle \text{Ц}, \Phi, \Pi_s, S, W_l, J, S_r(S') \rangle,$$

де:

– Ц – ціль ухвалення рішення;

– Ф – вихідні дані для породження варіантів набору засобів захисту:

$$\Phi = \{\Phi_1, \Phi_2, \ldots, \Phi_l, \ldots, \Phi_L\};$$

– Π_s – правило породження варіантів набору, що може бути представлене в аналітичному виді як векторний добуток множин:

$$S = \Phi_1 \times \Phi_2 \times \ldots \times \Phi_l \times \ldots \times \Phi_L,$$

де Φ_l – множина, що складається із засобів захисту l-ої функціональної підсистеми:

$$\Phi_l = \{A_{l1}, A_{l2}, \ldots A_{lm} \ldots, A_{lK_l}\};$$

– S – множина породжених варіантів набору;

– W_l – дані для вибору раціональних варіантів;

– J – цільова функція для вибору раціонального набору засобів захисту(правило вибору);

– S_r – раціональний набір засобів захисту.

Відзначається, що в умовах автоматизованого управління й при використанні експертної інформації в процесі ухвалення рішення можна говорити (навіть у випадку формалізованого правила вибору) про раціональне, а не оптимальне рішенні.

Відповідно до трьохрубіжної моделі захисту, основою планування раціонального модульного состава СЗІ є функціональні вимоги до наборів ЗЗ для кожного рубежу, які формулюються на основі нормативної документації, відповідно до рівня критичності оброблюваної інформації. Альтернативні засоби захисту для кожної функціональної підсистеми набору засобів захисту вибираються з урахуванням цих вимог. Варіантів наборів, сертифікованих по необхідному класі захищеності, може бути багато. Порівняння варіантів наборів засобів захисту пропонується робити по кількісній мері.

Для рішення завдання вибору раціональних варіантів наборів засобів захисту для рубежів захисту розробляється метод обробки знань, що використовує неформалізуємий досвід експерта в області ЗІ, що забезпечує перетворення відомостей про характеристики засобів захисту з бази знань і вивід

рішення в аналітичній формі – метод формування раціонального комплексу засобів захисту для СЗІ.

1. Розробляються варіанти набору ЗЗ. Множина можливих варіантів рішення завдяня вибору задається морфологічною матрицею. Розробляються морфологічні матриці засобів захисту для трьох рубежів.

2. Заповнюються допоміжні матриці, у яких відзначаються сумісні один з одним програмно-апаратні засоби. Допоміжна квадратна матриця сумісних рішень заповнюється в такий спосіб: для кожної пари засобів захисту різних функціональних підсистем визначається, чи сумісні вони, і результат заноситься в таблицю. Якщо ЗЗ сумісні, то функція сумісності $s(A_{lm}, A_{pr}) = 1$, у противному випадку $s(A_{lm}, A_{pr}) = 0$.

3. Генерується безліч рішень на вибір варіантів набору ЗЗ із усіканням цієї множини до підмножини варіантів набору із сумісних між собою програмно-апаратних продуктів.

Множина $S = \{S_1, ...S_r, ..., S_R\}$, що складається із всіх можливих варіантів побудови набору ЗЗ для рубежу, є декартовим добутком множин альтернатив (рядків морфологічної матриці).

Елемент множини:

$$S_r = \left\{ \left(A_{1i}, A_{2j}, \dots A_{lm}, \dots A_{Ln} \right) : A_{lm} \in \Phi_l, \forall l = \overline{1, L} \right\},$$

де:

– L – число функціональних підсистем для рубежу;

– A_{lm} – засіб захисту для реалізації l-ой функціональної підсистеми.

Генерація множин рішень на вибір варіантів набору, що складаються із сумісних між собою ЗЗ, здійснюється в такий спосіб.

Відбувається ітераційний синтез варіантів набору, що складаються із сумісних ЗЗ: на першому кроці перебираються послідовно варіанти засобів захисту для першої підсистеми, після вибору альтернативи A_{1i} здійснюється перехід до другого кроку. На другому кроці виконується послідовний перебір

варіантів засобів захисту другої підсистеми, але вибір здійснюється тільки для таких альтернатив A_{2j}, для яких функція сумісності $s(A_{1i}, A_{2j}) = 1$ і т.д. При виборі альтернатив з l-ої підсистеми вибір здійснюється тільки з таких альтернатив A_{lm}, для яких функції сумісності дорівнюють одиниці: $s(A_{l-1,m}, A_{lm}) = 1$, $s(A_{2j}, A_{lm}) = 1$, $s(A_{1i}, A_{lm}) = 1$... Таким, образом, вибір ЗЗ із кожного рядка морфологічної матриці (по одній з кожного рядка) для формування варіанта набору здійснюється тільки із сумісних між собою програмно-апаратних продуктів.

4. Подальше усікання множини S виконується методом повного перебору по заданій цільовій функції. Як цільова функція для вибору варіанта набору $S_r = \{A_{1i}, A_{2j}, \ldots A_{lm}, \ldots A_{Ln}\}$, застосовується функція:

$$J = \max_r \frac{W_{K^1_{\text{зщ}}}^{A_{1i}} + \ldots + W_{K^1_{\text{зщ}}}^{A_{lm}} + \ldots + W_{K^1_{\text{зщ}}}^{A_{Ln}}}{W_{K^1_{\text{и}}}^{A_{1i}} + \ldots + W_{K^1_{\text{и}}}^{A_{lm}} + \ldots + W_{K^1_{\text{и}}}^{A_{Ln}}},$$

де:

– $W_{K^1_{\text{зщ}}}^{A_{lm}}$ – значення показника «захищеність»;

– $W_{K^1_{\text{и}}}^{A_{lm}}$ – значення показника «витрати» засобу захисту A_{lm}.

Для оцінки засобів захисту різних функціональних підсистем наборів розробляються ієрархічні структури узагальнених критеріїв якості засобів захисту: показник «захищеність» і показник «витрати».

Критерії якості засобів захисту по ієрархії «захищеність» діляться на дві групи: показники забезпечення ефективності оперативних методів захисту й показники функціональної придатності. Критерії якості по ієрархії «витрати» діляться також на дві групи: у першу включена вартість відповідного засобу захисту, число користувачів по однієї ліцензії й інші можливі економічні витрати; до другої групи витрат ставляться функціональні витрати, такі, наприклад, як падіння продуктивності інформаційної системи при використанні даного засобу захисту.

Оцінка засобів захисту й критеріїв здійснюється попарним порівнянням по методу Т. Сааті, результати приводяться в числовому виді. З використанням ієрархічних структур критеріїв якості ЗЗ обчислюються нормовані значення власних векторів засобів захисту за всіма критеріями до показників «захищеність» $K^1_{\text{зщ}}$ і «витрати» $K^1_{\text{и}}$ на підставі обробки всіх матриць попарних порівнянь із урахуванням зв'язків критеріїв.

Після вибору раціональних наборів засобів захисту для рубежів захисту отриманий раціональний модульний состав цілісного комплексу засобів захисту об'єкта, що задовольняє вимозі

$$J \rightarrow \max.$$

5. Оцінюється, чи задовольняє сформований комплекс засобів захисту вимозі:

$$C_{\Sigma} \le C$$

де:

– C_{Σ} – сумарні витрати на реалізацію комплексу ЗЗ;

– $C_{\text{доп}}$ – виділені на реалізацію комплексу грошові ресурси.

При цьому C_{Σ} обчислюється за допомогою наступного вираження:

$$C_{\Sigma} = \sum_{S} \left(\sum i_S + \sum C^{\text{с}}_{j_S} + \sum C^{\text{в}}_{k_S} + C_{\text{сегм}} \right) + C_{\text{пр}},$$

де:

– S – число мережних сегментів;

– $C^{\text{Б}}_{i_S}$ – вартість набору засобів захисту хоста, на якому обробляється інформація базового рівня критичності;

– $C^{\text{С}}_{j_S}$ – вартість набору засобів захисту хоста, на якому обробляється інформація середнього рівня критичності;

– $C^{\text{В}}_{k_S}$ – вартість набору засобів захисту хоста, на якому обробляється інформація високого рівня критичності;

– $C_{\text{сегм}_s}$ – вартість набору засобів захисту на границі *s*-го мережного сегмента;

– $C_{\text{пр}}$ – вартість наборів засобів захисту периметра.

Вибір комплексу засобів захисту для СЗІ досягається ітераційно шляхом наближення до раціонального состава, що задовольняє вимогам до припустимих витрат на його реалізацію.

У системі інтелектуальної підтримки раціональні рішення пропонується вибирати на основі використання експертних знань; у ній реалізується механізм придбання знань у процесі заповнення полів знань експертом при взаємодії його з автоматизованою системою, виконується сукупність процедур над проблемною областю з використанням багатокритеріального порівняльного аналізу для виявлення в заданому експертом множини підмножини найкращих за критеріями переваги варіантів наборів, з яких формується раціональний комплекс засобів захисту.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

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.

Діаграми потоків даних містять чотири типи елементів:

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

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

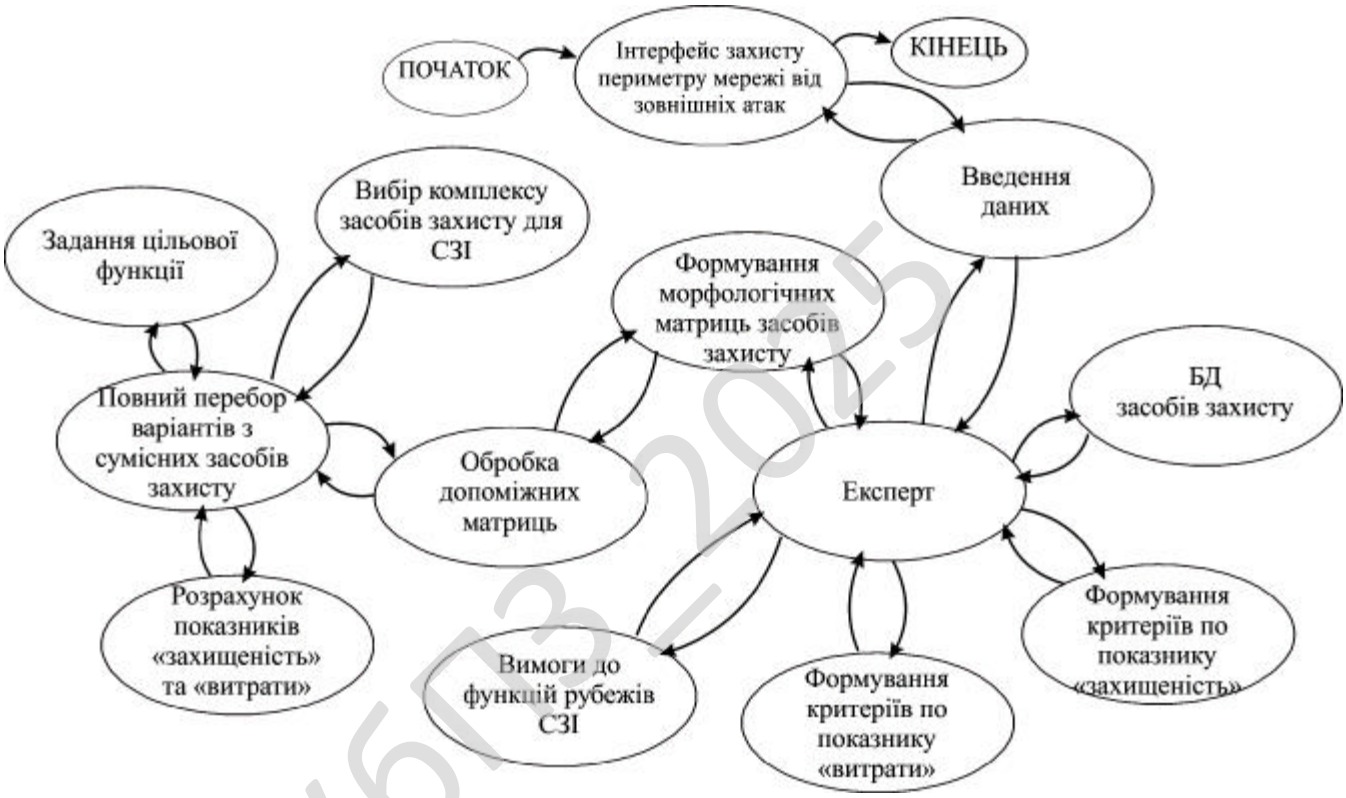

Рисунок 3.3 – Діаграма взаємодії процесів

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

Блок-схеми є першоджерелами стратегії розвитку ПЗ. 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 оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю захисту периметру мережі від зовнішніх атак.

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем. 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 підпрограми 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.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

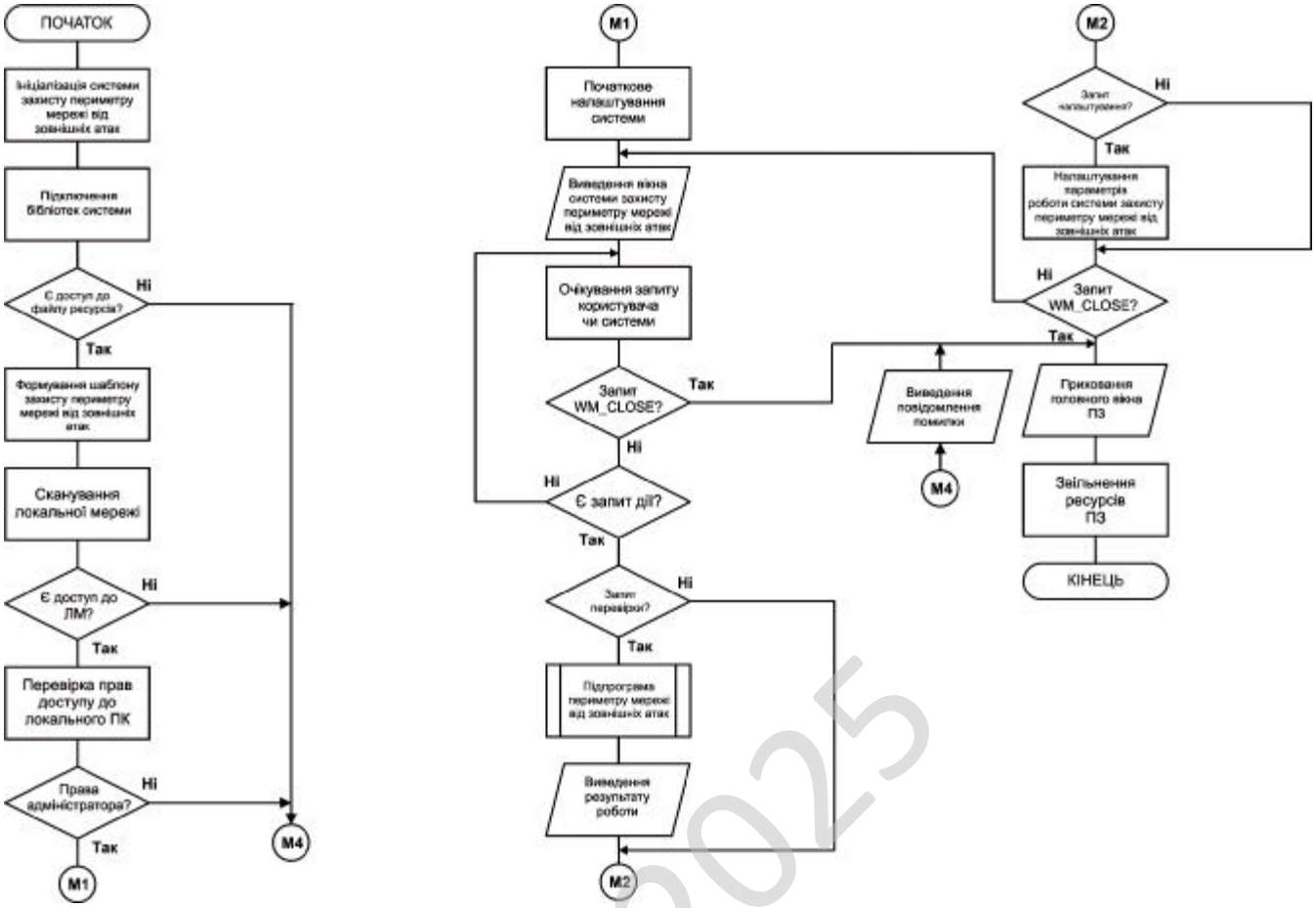

Рисунок 4.1 – Блок-схема основної програми

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

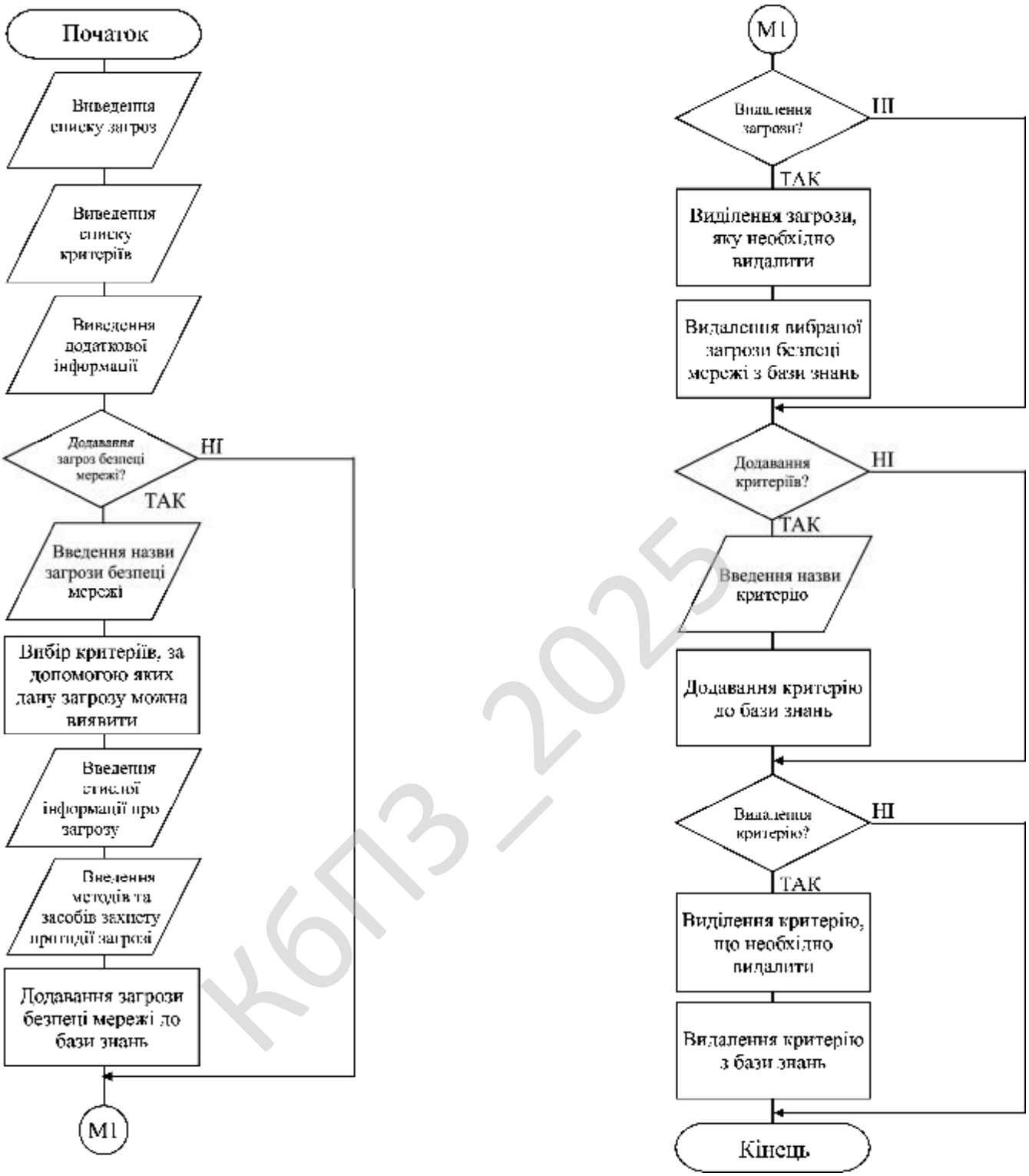

Рисунок 4.2 – Блок-схема роботи підпрограми

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Також при розробці бакалаврської дипломної роботи було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу); Діаграма класів; Діаграма компонент.

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.

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.

Діаграма класів це статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення.

Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм.

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

В UML існують наступні типи зв'язків які використовуються у діаграмі класів: Асоціації; Агрегація; Композиція.

Асоціації це якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі випадки, коли обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу дозволено зв'язати інші об'єкти з того ж класу. Асоціація, що зв'язує два класи, називається бінарної. Можна, хоча це рідко буває необхідним, створювати асоціації, що зв'язують відразу кілька класів. Графічно асоціація зображується у вигляді лінії, що з'єднує клас сам з собою або з іншими класами.

Асоціації може бути присвоєно ім'я, яке описує природу відносини. Зазвичай ім'я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає необхідність посилатися на них і відрізняти один від одного. Ім'я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій.

Клас, що бере участь в асоціації, грає в ній деяку роль. По суті, це "обличчя", яким клас, що знаходиться на одній стороні асоціації, звернений до класу з іншого її боку. Можна явно позначити роль, яку клас грає в асоціації.

Часто при моделюванні буває важливо вказати, скільки об'єктів може бути пов'язано допомогою одного примірника асоціації. Це число називається кратністю (Multiplicity) ролі асоціації та записується або як вираз, значенням якого є діапазон значень, або в явному вигляді.

Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати діапазон: "нуль або одиниця" (0..1), "багато" (0 .. *), "одиниця або більше" (1 .. *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає "будь-яке число об'єктів, крім 2 і 5".

Агрегація це проста асоціація між двома класами відображає структурний відношення між рівноправними сутностями, коли обидва класу знаходяться на одному концептуальному рівні і ні один не є більш важливим, ніж інший. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин).

Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і зображується у вигляді простої асоціації з незафарбованим ромбом з боку «цілого». Графічно агрегація представляється порожнім ромбом на блоці класу, і лінією, яка від цього ромба до міститься класу.

Композиція це більш суворий варіант агрегації. Відома також як агрегація за значенням.

Композиція має жорстку залежність часу існування екземплярів класу контейнера та примірників містяться класів. Якщо контейнер буде знищений, то весь його вміст буде також знищено. Графічно представляється як і агрегація, але з зафарбовані ромбиком.

Діаграма компонент в UML це діаграма, на якій відображаються компоненти, залежності та зв'язки між ними.

Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись.

Модуль програмного забезпечення може бути представлено в якості компоненти. Деякі компоненти існують під час компіляції, деякі – під час компонування, а деякі під час роботи програми.

Діаграма компонент відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгортування.

Компоненти об'єднуються разом використовуючи структурні зв'язки (assembly connector) щоб об'єднати інтерфейси двох компонент. Це ілюструє зв'язок типу «клієнт-сервер».

Структурна взаємодія – «зв'язок двох компонент, який передбачає, що один з них надає послуги, потрібні іншому компоненту».

При використанні діаграми компонент щоб показати внутрішню структуру компонента, клієнтські та серверні інтерфейси можуть утворювати пряме з'єднання з внутрішніми. Таке з'єднання називається з'єднанням делегації.

Нижче наведемо частину вихідного коду.

```
import scapy.all as scapy
import threading
import time
import os

# Ініціалізація класу для моніторингу та захисту периметру мережі
class PerimeterSecurity:

# Ініціалізація параметрів класу
    def __init__(self, interface="eth0"):
        self.interface = interface
        self.alerts = []
```

```
# Функція для виявлення ARP-спуфінгу
    def detect_arp_spoofing(self, packet):
        if packet.haslayer(scapy.ARP) and packet[scapy.ARP].op == 2:
            real_mac = self.get_mac(packet[scapy.ARP].psrc)
            response_mac = packet[scapy.ARP].hwsrc
            if real_mac and real_mac != response_mac:
                alert_msg = f"ARP Spoofing detected:
                            {packet[scapy.ARP].psrc} - {response_mac}"
                self.log_alert(alert_msg)

# Функція для отримання MAC-адреси за IP
    def get_mac(self, ip):
        arp_request = scapy.ARP(pdst=ip)
        broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
        arp_request_broadcast = broadcast / arp_request
        answered_list = scapy.srp(arp_request_broadcast, timeout=1,
                        verbose=False)[0]
        if answered_list:
            return answered_list[0][1].hwsrc
        return None

# Функція для журналювання подій безпеки
    def log_alert(self, message):
        timestamp = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())
        log_message = f"{timestamp} - {message}"
        self.alerts.append(log_message)
        with open("security_log.txt", "a") as log_file:
            log_file.write(log_message + "\n")

# Функція для моніторингу мережевого трафіку
    def monitor_network(self):
        scapy.sniff(iface=self.interface, store=False,
                    prn=self.detect_arp_spoofing)

# Функція для блокування атакуючих IP-адрес
    def block_ip(self, ip):
        os.system(f"iptables -A INPUT -s {ip} -j DROP")
        alert_msg = f"Blocked IP: {ip}"
        self.log_alert(alert_msg)

# Запуск системи моніторингу у фоновому потоці
    def start_monitoring(self):
```

```
            monitor_thread = threading.Thread(target=self.monitor_network,
                                              daemon=True)
            monitor_thread.start()

    # Ініціалізація захисту периметру
    security_system = PerimeterSecurity()
    security_system.start_monitoring()

    # Вивід повідомлення про запуск системи
    print("Network perimeter security system is active.")
```

Оцінювання рівня захищеності інформації здійснюється на основі одного з основних положень уніфікованої концепції захисту – вимоги науково обґрунтованого підходу до оцінки (бажано в кількісному вираженні) необхідного рівня захищеності при проектуванні й у процесі експлуатації СЗІ.

У процесі аналізу й оцінювання ризиків установлюється ступінь адекватності використовуваних або планованих наборів засобів захисту (ЗЗ) існуючим погрозам. Властивість «захищеність інформації» кожного ЗЗ, що входить у СЗІ, у сукупності визначає захищеність інформації в СЗІ в цілому. Наявність уразливості ЗЗ може привести до порушення захищеності, тобто здійсненню погрози, тому при рішенні завдань захисту інформації першорядне значення має кількісна оцінка уразливістей засобів захисту. Оскільки вплив на інформацію різних деструктивних факторів значною мірою є випадковим, то як кількісну міру уразливості найбільше доцільно застосувати ймовірність порушення захищеності інформації.

Неясність способу визначення значень ймовірностей погроз і уразливістей є основною проблемою при одержанні кількісної оцінки ризику порушення інформаційної безпеки. Відомо, що застосування методів класичної теорії ймовірностей припустимо при повторюваності дослідів і однаковості умов. Це вимога в складних системах, якими є СЗІ, звичайно не виконується. Відповідно до одному із принципів системного аналізу – принципу невизначеності, у процесі дослідження системи необхідний облік невизначеностей і випадків. Оскільки складні відкриті системи не підкоряються імовірнісним законам, у них варто

оцінити найгірші ситуації відповідно до методу гарантованого результату, що пропонується використовувати при оцінці ймовірностей погроз.

Приймається, що значення показника *m*-го ЗЗ захищеність інформації $P_{бm}$ – це суб'єктивна ймовірність виявлення й блокування засобом захисту несанкціонованих дій, тобто теоретична очікувана ефективність бар'єра.

Очевидно, що ймовірність порушення захищеності $P^{н}_{бm}$ доповнює $P_{бm}$ до одиниці, тобто:

$$P^{н}_{бm} = 1 - P_{бm} ,$$

де $P^{н}_{бm}$ – імовірність порушення захищеності інформації, або ймовірність уразливості *m*-го ЗЗ (імовірність подолання бар'єра).

Пропонується ймовірносно-статичний підхід, при якому не враховується динаміка зміни значень ймовірностей погроз і уразливістей у часі, оцінюються апріорні очікувані значення ймовірностей порушення захищеності інформації.

Особливістю пропонованого в магістерській роботі підходу є одержання чисельних значень суб'єктивних ймовірностей на основі використання як приватні показники захищеності технічних характеристик і можливостей засобів захисту, декларуємих розроблювачами. Вирішується завдання одержання чисельної оцінки узагальненого показника якості засобу захисту.

Пропонується для одержання чисельної оцінки узагальненого показника якості засобу захисту захищеність інформації використовувати теорію нечітких множин. Для оцінки засобів захисту за кожним критерієм нижнього ієрархічного рівня формуються функції приналежності. При цьому використовуються методи побудови функцій приналежності, засновані на формалізації й інтеграції нечітких даних, сформованих експертом у процесі оцінювання параметрів реальних засобів захисту. Формулюються відповідні продукційні правила, що дозволяють обробляти складні з'єднання. Достоїнство способу – відносно висока об'єктивність.

Метод оцінювання рівня захищеності інформації базується на трьохрубіжній моделі захисту, він розроблений для об'єкта захисту, архітектура якого відповідає основним принципам безпеки, що рекомендуються.

Відомо, що рівень захищеності й відносний ризик доповнюють один одного до одиниці. Пропонується розраховувати рівень захищеності η за формулою:

$$\eta = 1 - \overline{R} = 1 - \sum_s \frac{C_s}{C_\Sigma} \cdot P_s,$$

де $\overline{R}$ – відносний ризик;

C_s – частка вартості інформаційних ресурсів, що захищаються, у сегменті s;

s – номер сегмента;

S – число сегментів;

P_s – результуюча ймовірність погроз інформаційному середовищу сегмента;

C_Σ – сумарний неприйнятний збиток;

$\frac{C_s}{C_\Sigma}$ – коефіцієнт небезпеки сукупності погроз в s-ому сегменті, обумовлений як частка вартості інформації, що захищається, об'єкта захисту, оброблюваного в сегменті.

Таким чином, для оцінки рівня захищеності потрібна кількісна оцінка ймовірностей реалізації каналів несанкціонованого доступу.

Для оцінки ймовірності порушення захищеності підмножиною порушників $\{K'\}$ по підмножині можливих каналів несанкціонованого одержання інформації $\{J'\}$ для сегмента s використовується співвідношення:

$$P_{s\{J'\}\{K'\}} = 1 - \prod_{J'} \left(1 - P_{sjk}^{(б)}\right) \prod_{K'} \left(1 - P_{sjk}^{(б)}\right),$$

у якому приймається:

$$P_{sjk}^{(б)внш} \subset P_{sjk}^{(б)}, P_{sjk}^{(б)вн} \subset P_{sjk}^{(б)},$$

де $P_{sjk}^{(б)вн}, P_{sjk}^{(б)внш}$ – імовірність несанкціонованого одержання інформації, оброблюваної в *s*-му сегменті, відповідно, внутрішнім і зовнішнім порушником (зловмисником) для об'єкта захисту, що має точки виходу в глобальну мережу, зовнішні виділені канали зв'язку, для якого можливі віддалені атаки через периметр.

З обліком прийнятої трьохрубіжної моделі захисту $P_{sjk}^{(б)внш}$ обчислюється за формулою:

$$P_{sjk}^{(б)внш} = 1 - \prod_{l=1}^{3} (1 - P_{sjk\ l}^{внш}),$$

де $P_{sjk\ l}^{внш}$ – імовірність несанкціонованого одержання інформації, оброблюваної в *s*-му сегменті, зловмисника, зовнішнього порушника у випадку подолання відповідного рубежу захисту *l*.

Імовірність $P_{sjk\ l}^{внш}$ залежить від чотирьох факторів і визначається залежністю:

$$P_{sjkl}^{внш} = P_{skl}^{д} \cdot P_{sjkl}^{н} \cdot P_{sjl}^{к} \cdot P_{sjl}^{и},$$

де $P_{sk\ l}^{д}$ – імовірність спроби доступу зловмисника або зовнішнього порушника-користувача до *l*-му рубежу захисту;

$P_{sjk\ l}^{н}$ – імовірність подолання зловмисником або зовнішнім порушником *l*-го рубежу захисту;

$P_{sj\ l}^{к}$ – імовірність наявності трафіка із сегмента s через l-*й* рубіж захисту, залежить від технології обробки інформації на об'єкті захисту, імовірність можна прийняти рівній частоті роботи каналу;

$P_{sj\ l}^{и}$ – імовірність наявності інформації, що захищається, *s*-го сегмента в трафіку в момент подолання зовнішнім порушником *l*-го рубежу захисту, залежить від технології обробки інформації на об'єкті захисту.

Внутрішній порушник у процесі реалізації каналів несанкціонованого доступу повинен перебороти два рубежі захисту.

Тоді ймовірність несанкціонованого одержання інформації, оброблюваної в сегменті s, внутрішнім порушником обчислюється за формулою:

$$P_{sj}^{(б)вн} = 1 - \prod_{l=1}^{2} (1 - P_{sjl}^{вн}),$$

де $P_{sl}^{вн}$ – імовірність несанкціонованого доступу до інформації, оброблюваної в *s*-му сегменті, внутрішнього порушника у випадку подолання відповідного рубежу захисту *l*. З перерахованих ймовірностей, що входять у формули для розрахунку $P_{sjl}^{вн}$ й $P_{sjkl}^{внш}$, одна з ймовірностей, а саме $P_{sjkl}^{н}$, залежить від якості використовуваних у системі засобів захисту й кількості бар'єрів на рубежі захисту. Якщо порушникові необхідно перебороти *M* бар'єрів на рубежі захисту, то ймовірність його вдалої атаки визначається як добуток:

$$P_{sjkl}^{н} = \prod_{m=1}^{M} P_{бm}^{н} = \prod_{m=1}^{M} (1 - P_{бm}).$$

4.2 Захист розробленого програмного забезпечення

Дані в програмі захищаються за допомогою використання алгоритму CAST-128 (або CAST5) у криптографії, це блоковий алгоритм симетричного шифрування на основі мережі Фейстеля, який використовується в цілому ряді продуктів криптографічного захисту, зокрема деяких версіях PGP і GPG і крім того схвалений для використання Канадським урядом.

Основні відомості

Алгоритм був створений в 1996 році Карлайлом Адамсом (Carlisle Adams) і Стаффордом Таваресом (Stafford Tavares) використовуючи метод побудови шифрів CAST, який використовується також і іншим їхнім алгоритмом CAST-256 (алгоритм-кандидат AES).

CAST-128 складається з 12 або 16 раундів мережі Фейстеля з розміром блоку 64 біта й довжиною ключа від 40 до 128 біт (але тільки з інкрементацією по 8 біт). 16 раундів використовуються коли розміри ключа перевищують 80 біт. В алгоритмі використовуються 8x16 S-блоки, засновані на бент-функції, операції XOR і модулярной арифметиці (медулярне додавання й вирахування). Є три різні типи функцій раундів, але вони схожі за структурою й різняться тільки у виборі виконуваної операції (додавання, вирахування або XOR) у різних місцях.

Хоча CAST-128 захищений патентом Entrust, його можна використовувати в усьому світі для комерційних або некомерційних цілей безкоштовно.

Опис

CAST – це популярний 64-бітовий шифр, що допускає розміри ключа аж до 128 біт. Алгоритм CAST використовує 64-бітовий блок і 64-бітовий ключ. CAST стійкий до диференціального й лінійного криптоаналізу. Сила алгоритму CAST укладена в його S-блоках. В CAST немає фіксованих S-блоків і для кожного додатка вони конструюються заново. Створений для конкретної реалізації CAST S-блок уже більше ніколи не міняється. Інакше кажучи, S-блоки залежать від реалізації, а не від ключа. Northern Telecom використовує CAST у своєму пакеті програм Entrust для комп'ютерів Macintosh, PC і робочих станцій UNIX. Обрані ними S-блоки не опубліковані, що втім не дивно.

CAST-128 належить компанії Entrust Technologies, але є безкоштовним як для комерційного, так і для некомерційного використання. CAST-256 – безкоштовне доступне розширення CAST-128, яке ухвалює розмір ключа до 256 біт і має розмір блоку 128 біт. CAST-256 був одним з первісних кандидатів на AES.

Опис алгоритму

CAST-128 заснований на мережі Фейстеля. Повний алгоритм шифрування викладений у наступних чотирьох кроках:

ВХІД: текст $m_1 \dots m_{64}$, ключ $K = k_1 \dots k_{128}$.

ВИХІД: зашифрований текст $c_1 \dots C_{64}$.

1. (розгорнення ключа) становить 16 пару підключів $\{Km_i, Kr_i\}$ отриманих з K (див. розділи Пари раундових ключів і Неідентичні раунди).

2. (L_0, R_0) <- $(m_1 .. m_{64})$. (Розділяє текст на ліву й праву 32-бітні половини $L_0 = m_1 ... m_{32}$ і $R_0 = m_{33} ... m_{64}$).

3. (16 раундів) for i from 1 to 16, обчислити L_i і R_i у такий спосіб: $L_i = R_{i-1}$; $R_i = L_{i-1}$ ^ $F(R_{i-1},Km_i,Kr_i)$, де F визначена в розділі «Пари раундових ключів» (F має тип 1, тип 2, тип 3 або, залежно від i).

4. $c_1 ... c_{64}$ <- (R_{16}, L_{16}). (Міняємо остаточні блоки місцями L_{16}, R_{16} і поєднуємо, щоб сформувати зашифрований текст.)

Розшифруванні збігається з алгоритмом шифрування, наведеним вище, крім того, що раунди (i, отже, пари підключів), використовуються у зворотному порядку, щоб обчислити (L_0, R_0) з (R_{16}, L_{16}).

Пари раундових ключів

CAST-128 використовує пару підключів за раунд: 32-бітні величини Km використовується в якості "маскування" ключа й Kr використовують як "перестановки" ключа, з яких використовуються тільки початкові 5-біт.

Неідентичні раунди

Три різні типів функції використовуються в CAST-128. Типи виглядає в такий спосіб (де "D" є вхідними даними у функцію F і "Ia"-"Id" є найбільш значимий байт – найменш значимий байт I, відповідно). Зверніть увагу, що "+" і "-" додавання й вирахування по модулю 2 ** 32, "" є побітове XOR і "<<<" є циклічним зрушенням уліво.

Раунди	$I = ((Km_i + R_{i-1}) <<< Kr_i)$
1,4,7,10,13,16	$F = ((S1[I_a]\ \ S2[I_b]) – (S3[I_c]))+ S4[I_d]$
Раунди	$I = ((Km_i \wedge R_{i-1}) <<< Kr_i)$
2,5,8,11,14	$F = ((S1[I_a] – S2[I_b]) + (S3[I_c]))\ \ S4[I_d]$
Раунди	$I = ((Km_i – R_{i-1}) <<< Kr_i)$
3,6,9,12,15	$F = ((S1[I_a] + S2[I_b])\ \ (S3[I_c]))- S4[I_d]$

Поля заміни

CAST-128 використовує вісім полів заміни: поля S1, S2, S3 і S4 раундові функції полів заміни, S5, S6, S7 і S8 є ключами розгорнення полів заміни. Незважаючи на те, що 8 полів заміни вимагають у цілому 8 Кбайт для зберігання, зверніть увагу на те, що тільки 4 Кбайта потрібні під час фактичного шифрування / дешифрування, тому що генерація підключа звичайно робиться до будь-якого введення даних.

Ключі розгорнення

Представимо 128-розрядний ключ у вигляді x0x1x2x3x4x5x6x7x8x9xaxbxcxdxexf, де x0 старший байт, і xf молодший байт.

Представимо z0..zf проміжними (тимчасовими) байтами. Si[] представляє поле заміни і і " ^ " представляє додавання по Xor'у.

Поля заміни формуються із ключа x0x1x2x3x4x5x6x7x8x9xaxbxcxdxexf у такий спосіб.

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K1 = S5[z8] ^ S6[z9] ^ S7[z7] ^ S8[z6] ^ S5[z2]

K2 = S5[zA] ^ S6[zB] ^ S7[z5] ^ S8[z4] ^ S6[z6]

K3 = S5[zC] ^ S6[zD] ^ S7[z3] ^ S8[z2] ^ S7[z9]

K4 = S5[zE] ^ S6[zF] ^ S7[z1] ^ S8[z0] ^ S8[zC]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K5 = S5[x3] ^ S6[x2] ^ S7[xC] ^ S8[xD] ^ S5[x8]

K6 = S5[x1] ^ S6[x0] ^ S7[xE] ^ S8[xF] ^ S6[xD]

K7 = S5[x7] ^ S6[x6] ^ S7[x8] ^ S8[x9] ^ S7[x3]

K8 = S5[x5] ^ S6[x4] ^ S7[xA] ^ S8[xB] ^ S8[x7]

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K9 = S5[z3] ^ S6[z2] ^ S7[zC] ^ S8[zD] ^ S5[z9]

K10 = S5[z1] ^ S6[z0] ^ S7[zE] ^ S8[zF] ^ S6[zC]

K11 = S5[z7] ^ S6[z6] ^ S7[z8] ^ S8[z9] ^ S7[z2]

K12 = S5[z5] ^ S6[z4] ^ S7[zA] ^ S8[zB] ^ S8[z6]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K13 = S5[x8] ^ S6[x9] ^ S7[x7] ^ S8[x6] ^ S5[x3]

K14 = S5[xA] ^ S6[xB] ^ S7[x5] ^ S8[x4] ^ S6[x7]

K15 = S5[xC] ^ S6[xD] ^ S7[x3] ^ S8[x2] ^ S7[x8]

K16 = S5[xE] ^ S6[xF] ^ S7[x1] ^ S8[x0] ^ S8[xD]

половина, що залишається, ідентична тому, що дане вище, продовження від останнього створило x0..xf, щоб генерувати ключі K17 – K32.

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K17 = S5[z8] ^ S6[z9] ^ S7[z7] ^ S8[z6] ^ S5[z2]

K18 = S5[zA] ^ S6[zB] ^ S7[z5] ^ S8[z4] ^ S6[z6]

K19 = S5[zC] ^ S6[zD] ^ S7[z3] ^ S8[z2] ^ S7[z9]

K20 = S5[zE] ^ S6[zF] ^ S7[z1] ^ S8[z0] ^ S8[zC]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K21 = S5[x3] ^ S6[x2] ^ S7[xC] ^ S8[xD] ^ S5[x8]

K22 = S5[x1] ^ S6[x0] ^ S7[xE] ^ S8[xF] ^ S6[xD]

K23 = S5[x7] ^ S6[x6] ^ S7[x8] ^ S8[x9] ^ S7[x3]

K24 = S5[x5] ^ S6[x4] ^ S7[xA] ^ S8[xB] ^ S8[x7]

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K25 = S5[z3] ^ S6[z2] ^ S7[zC] ^ S8[zD] ^ S5[z9]

K26 = S5[z1] ^ S6[z0] ^ S7[zE] ^ S8[zF] ^ S6[zC]

K27 = S5[z7] ^ S6[z6] ^ S7[z8] ^ S8[z9] ^ S7[z2]

K28 = S5[z5] ^ S6[z4] ^ S7[zA] ^ S8[zB] ^ S8[z6]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K29 = S5[x8] ^ S6[x9] ^ S7[x7] ^ S8[x6] ^ S5[x3]

K30 = S5[xA] ^ S6[xB] ^ S7[x5] ^ S8[x4] ^ S6[x7]

K31 = S5[xC] ^ S6[xD] ^ S7[x3] ^ S8[x2] ^ S7[x8]

K32 = S5[xE] ^ S6[xF] ^ S7[x1] ^ S8[x0] ^ S8[xD]

Маскування й перестановка підключів

Km_1, ..., Km_{16} 32-розрядні підключи маскування (один на раунд).

Kr_1, …, Kr_{16} 32-розрядні перестановки підключів (один на раунд); тільки молодші 5 бітів використовуються в кожному раунді.

for (i=1; i<=16; i++) { Km_i = Ki; Kr_i = K16+i; }

Змінний розмір ключа

CAST-128 Алгоритм шифрування був розроблений, щоб розмір ключа міг варіюватися від 40 до 128 біт, в 8-бітному кроці (тобто припустимі розміри ключа рівняються 40, 48, 56, 64..., 112, 120, і 128 бітам). Для змінної роботи розміру ключа специфікація наступні:

1) Для розмірів ключа до й включаючи 80 бітів (тобто, 40, 48, 56, 64, 72, і 80 бітів) алгоритм точно такої ж, але використовує 12 раундів замість 16;

2) Для розмірів ключа більше, чим 80 бітів, алгоритм використовує повні 16 раундів;

3) Для розмірів ключа менше, чим 128 бітів ключ доповнений нульовими байтами (у самих правих, або молодших, позиціях) до 128 биток (тому що розклад ключа CAST 128 ухвалює вхідний ключ 128 бітів).

Розшифрування

Розшифрування для CAST-128 відносно проста. Розшифрування працює в тому ж алгоритмічному напрямку, що й шифрування, починаючи із зашифрованого тексту як вхідних даних. При цьому підключ використовуються у зворотному напрямку.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено вікно додавання загрози захисту периметру мережі, розробленого у результаті виконання бакалаврської дипломної роботи. Розроблене програмне забезпечення захисту периметру мережі від зовнішніх атак нема головного вікна ПЗ, воно працює автоматично у вигляді сервісу та при необхідності виводить вікно додавання нової загрози яка складається з наступних функціональних блоків:

– Блоку додавання загрози який складається з: Критерію; Введеної назви загрози; Стислий опис загрози; Критеріїв виявлення загрози; Методів та засобів протидії загрозам безпеки мережі.

– Блок формування загрози.

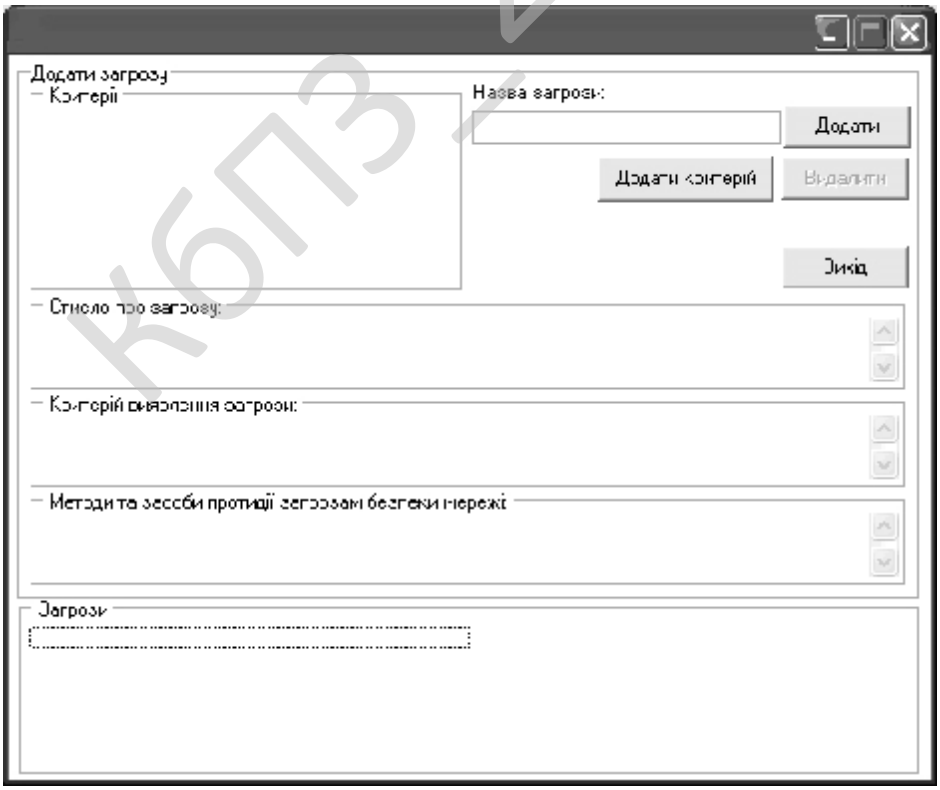

Рисунок 5.1 – Вікно додавання загрози захисту периметру мережі

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку авторського права, після чого на екрані з’явиться вікно показане на рисунку 5.2.

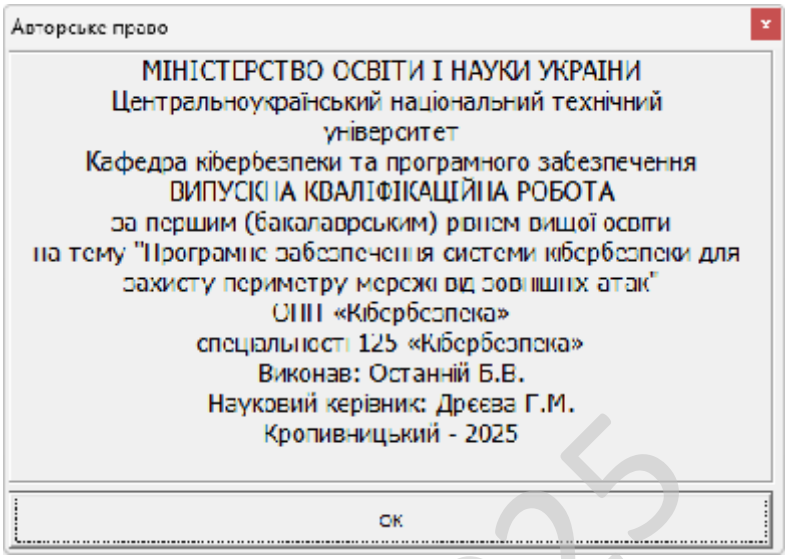

Рисунок 5.2 – Вікно розробника ПЗ

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 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 – Freeware.

Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів.

Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення.

Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки для захисту периметру мережі від зовнішніх атак.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем для захисту периметру мережі від зовнішніх атак.

– Досліджена система для захисту периметру мережі від зовнішніх атак.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки для захисту периметру мережі від зовнішніх атак.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання для захисту периметру мережі від зовнішніх атак.

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 Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки для захисту периметру мережі від зовнішніх атак. 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 кібербезпеки й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм CAST-128.

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.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Lakhno, V., Malyukov, V., Smirnov, O., Bebeshko, B., Chubaievskiy, V., Zhumadilova, M., Malyukova, I., Smirnov, S. «Multifactorial Model for Targeted Attacks Counteracting Within the Framework of a Multi-Step Quality Game with Fuzzy Information». *8th International Symposium on Intelligent Informatics, ISI 2023*, 2025. vol 389. pp 377-389. Springer, Singapore.

2. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

3. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447.

4. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

5. Akhalaia, G., Iavich, M., Iashvili, G., Prysiazhnyy, D., Smirnova, T. «Secure Encrypted Connection on Georgian Website». *CEUR Workshop Proceedings,* 2023, 3550, pp. 313-320.

6. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56

7. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

8. Kuznetsov, O., Kandiy, S., Frontoni, E., Smirnov, O. «Trade-offs in Post-Quantum Cryptography: A Comparative Assessment of BIKE, HQC, and Classic McEliece». *CEUR Workshop Proceedings,* Volume 3504, 2023, pp. 1-11.

9. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022,

10. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, vol 131. 2023. Springer, Singapore. pp. 21-34.

11. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. Springer, Cham. 2022, pp. 2463-2477.

12. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». *11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021,* Cracow, Poland, 22-25 September 2021. P. 414-418

13. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». *4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021*, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

14. Smirnov O., Kuznetsov A., Girzheva O., Kiian A., Nakisko O., Kuznetsova T. «Advanced Code-Based Electronic Digital Signature Scheme». *2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020,* Kharkiv, 6 October 2020-9 October 2020, P. 358-362.

15. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.*

16. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

17. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

18. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

19. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131.

20. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14.

21. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. Springer, Cham. 2021, pp 66-84.

22. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

23. Smirnov O., Kuznetsov A., Kiian A., Babenko V., Perevozova I.,Chepurko I. «New Approach to the Implementation of Post-Quantum Digital Signature Scheme». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 166-171.

24. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

25. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

26. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

27. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43.

28. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660.

29. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

30. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of

Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

31. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019.

32. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

33. Smirnov, O., Kuznetsov, A., Kiian, A., Gorbenko, Y., Cherep, O., Bexhter L. «Code-based Pseudorandom Generator for the Post-Quantum Period», *2019 IEEE International Conference on Advanced Trends in Information Theory (IEEE ATIT 2019).* 18.12.19-20.12.19 Kyiv Ukraine. P. 204 – 209.

34. Smirnov, O., Kuznetsov, A., Nariezhnii, O., Stelnyk, S., Kokhanovska, T., Kuznetsova T., «Side Channel Attack on a Quantum Random Number Generator», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18 - 21 September 2019. P.713-718.

35. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019;* Metz; France; 18-21 September 2019. P. 707-712.

36. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.701-706.

37. Smirnov, O., Hu, Z., Vasiliu, Y., Sydorenko, V., Polishchuk, Y., «Abstract Model of Eavesdropper and Overview on Attacks in Quantum Cryptography Systems», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.399-405.

38. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

39. Smirnov, O., Kuznetsov, A., Kiian, A., Babenko, B., Zhosan, H., Prokopovych-Tkachenko, D., «Soft Decoding Method for Turbo-Productive Codes», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019*, Lviv, Ukraine, 2-6 July, 2019, P. 129-134.

40. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

41. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

42. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

43. Smirnov, O., Kuznetsov, A., Kiian, A., Kuznetsova, K., Ivko, T., Prokopovych-Tkachenko, D., «Soft Decoding Based on Ordered Subsets of Verification Equations of Turbo-Productive Codes», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 873-884.

44. Smirnov A.A., Kuznetsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

45. Ткаченко, О., Ільєнко, А., Улічев, О., Мелешко, Є., Смірнов, О. «Правові засади поширення інформаційних впливів в соціальних мережах». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2024. № 2(26), С. 170–188.

46. Смірнова Т.В., Конопліцька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка.* 2024. №4(24), С. 6-27.

47. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка.* 2024. №3(23), С. 111-131.

48. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

49. Смірнова Т.В., Гнатюк С.О., Бердибаєв Р.Ш., Сидоренко В.М., Жигаревич О.К., «Система корелювання подій та управління інцидентами кібербезпеки на об’єктах критичної інфраструктури». *Кібербезпека: освіта, наука, техніка*, №*3*(19), 2023, С. 176-196.

50. Смірнов О.А. Козлов Я.О., Смірнова Т.В. «Дослідження застосування SIEM-систем для забезпечення кібербезпеки та захисту інформації». *ІІ Міжнародна науково-практична Інтернет-конференція «Інновації та перспективні шляхи hозвитку інформаційних технологій (ІПШРІТ-2023)»* м.Черкаси 6 грудня 2023 року – Черкаси: ЧДТУ.– 2023. – С.251-252.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки для захисту периметру мережі від зовнішніх атак.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 51-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки для захисту периметру мережі від зовнішніх атак.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) 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 кібербезпеки з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки для захисту периметру мережі від зовнішніх атак;

– 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, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Python.

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 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 71 аркуш.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 5.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

_____ Дрєєва Г.М.

Програмне забезпечення системи кібербезпеки для захисту периметру мережі від зовнішніх атак

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 24

Літера: РП

Кропивницький – 2025 року

Основна програма

```
#!/usr/bin/env python3
import socket
import threading
import time
import logging
import random
import re
import queue
import sys
import os
# Initialize logging configuration
logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s -
%(message)s')
# Define global configuration parameters for the cybersecurity system
CONFIG_SCAN_INTERVAL = 5  # seconds between network scans
CONFIG_ALERT_THRESHOLD = 3  # number of suspicious events before alert
CONFIG_FIREWALL_CHECK_INTERVAL = 10  # seconds for firewall rule recheck
# Define a list of simulated suspicious IP addresses
SUSPICIOUS_IPS = [
"192.168.1.101",
"10.0.0.203",
"172.16.0.45",
"192.168.1.150",
"10.0.0.205",
]
# Define a function to simulate network traffic packet generation
def generate_network_packet():
# Create a dummy network packet as a dictionary
# Packet includes source IP, destination IP, port, and payload
    src_ip = "192.168.1." + str(random.randint(1, 254))
    dst_ip = "10.0.0." + str(random.randint(1, 254))
    port = random.randint(1, 65535)
    payload = "DATA" + str(random.randint(1000, 9999))
    packet = {"src_ip": src_ip, "dst_ip": dst_ip, "port": port, "payload":
payload}
    return packet
# Define a class for Firewall functionalities
class Firewall:
# Initialize the firewall with empty rules
    def __init__(self):
        self.blocked_ips = set()
        self.allowed_ips = set()
        self.lock = threading.Lock()
# Add an IP address to the blocked list
    def add_block(self, ip_address):
        with self.lock:
            self.blocked_ips.add(ip_address)
            logging.info("Firewall: Blocked IP " + ip_address)
# Remove an IP address from the blocked list
    def remove_block(self, ip_address):
        with self.lock:
            if ip_address in self.blocked_ips:
                self.blocked_ips.remove(ip_address)
                logging.info("Firewall: Unblocked IP " + ip_address)
# Check if an IP address is blocked by the firewall
    def is_blocked(self, ip_address):
        return ip_address in self.blocked_ips
# Update firewall rules periodically based on detected threats
    def update_rules(self, suspicious_ip):
        if not self.is_blocked(suspicious_ip):
            self.add_block(suspicious_ip)
```

```
        logging.debug("Firewall: Updated rules with suspicious IP " +
suspicious_ip)
# Define a class for Intrusion Detection System functionalities
class IntrusionDetectionSystem:
# Initialize IDS with an event queue for suspicious events
    def __init__(self):
        self.suspicious_events = queue.Queue()
        self.event_counts = {}
        self.lock = threading.Lock()
# Analyze a network packet for suspicious activity
    def analyze_packet(self, packet):
        src_ip = packet.get("src_ip", "")
        payload = packet.get("payload", "")
        if re.search(r"DATA[0-9]{4}", payload):
            with self.lock:
                if src_ip in self.event_counts:
                    self.event_counts[src_ip] += 1
                else:
                    self.event_counts[src_ip] = 1
                if self.event_counts[src_ip] >= CONFIG_ALERT_THRESHOLD:
                    self.suspicious_events.put(src_ip)
                    logging.warning("IDS: Suspicious activity detected from " +
src_ip)
        logging.debug("IDS: Analyzed packet from " + src_ip)
# Retrieve suspicious events from the queue
    def get_suspicious_event(self):
        try:
            event = self.suspicious_events.get_nowait()
            return event
        except queue.Empty:
            return None
# Reset event counts periodically to prevent stale data
    def reset_event_counts(self):
        with self.lock:
            self.event_counts.clear()
            logging.info("IDS: Reset event counts")
# Define a class for Alert System functionalities
class AlertSystem:
# Initialize alert system with a log of alerts
    def __init__(self):
        self.alert_log = []
        self.lock = threading.Lock()
# Send an alert message to system administrator
    def send_alert(self, message):
        with self.lock:
            self.alert_log.append(message)
            logging.error("ALERT: " + message)
# Retrieve the complete alert log
    def get_alert_log(self):
        with self.lock:
            return list(self.alert_log)
# Define a class for Network Monitoring functionalities
class NetworkMonitor:
# Initialize network monitor with IDS and firewall components
    def __init__(self, ids, firewall):
        self.ids = ids
        self.firewall = firewall
        self.running = True
        self.monitor_thread = threading.Thread(target=self.monitor_network)
        self.lock = threading.Lock()
# Start the network monitoring process
    def start(self):
        self.monitor_thread.start()
```

```
        logging.info("NetworkMonitor: Started network monitoring thread")
# Stop the network monitoring process
    def stop(self):
        self.running = False
        self.monitor_thread.join()
        logging.info("NetworkMonitor: Stopped network monitoring thread")
# Monitor network traffic by generating and analyzing packets
    def monitor_network(self):
        while self.running:
            packet = generate_network_packet()
            if not self.firewall.is_blocked(packet.get("src_ip", "")):
                self.ids.analyze_packet(packet)
            else:
                logging.info("NetworkMonitor: Packet from blocked IP " +
packet.get("src_ip", "") + " ignored")
            time.sleep(CONFIG_SCAN_INTERVAL)
# Define a class to manage the entire cybersecurity system
class CyberSecuritySystem:
# Initialize the cybersecurity system with all components
    def __init__(self):
        self.firewall = Firewall()
        self.ids = IntrusionDetectionSystem()
        self.alert_system = AlertSystem()
        self.network_monitor = NetworkMonitor(self.ids, self.firewall)
        self.ids_processing_thread =
threading.Thread(target=self.process_ids_events)
        self.firewall_update_thread =
threading.Thread(target=self.firewall_rule_updater)
        self.running = True
# Start the cybersecurity system by starting all threads
    def start_system(self):
        self.network_monitor.start()
        self.ids_processing_thread.start()
        self.firewall_update_thread.start()
        logging.info("CyberSecuritySystem: All components started")
# Stop the cybersecurity system by stopping all threads
    def stop_system(self):
        self.network_monitor.stop()
        self.running = False
        self.ids_processing_thread.join()
        self.firewall_update_thread.join()
        logging.info("CyberSecuritySystem: All components stopped")
# Process suspicious events detected by IDS and send alerts
    def process_ids_events(self):
        while self.running:
            suspicious_ip = self.ids.get_suspicious_event()
            if suspicious_ip:
                self.firewall.update_rules(suspicious_ip)
                alert_message = "Suspicious activity detected from IP: " +
suspicious_ip
                self.alert_system.send_alert(alert_message)
            time.sleep(1)
# Periodically update firewall rules based on known suspicious IPs
    def firewall_rule_updater(self):
        while self.running:
            for ip in SUSPICIOUS_IPS:
                if not self.firewall.is_blocked(ip):
                    self.firewall.update_rules(ip)
            time.sleep(CONFIG_FIREWALL_CHECK_INTERVAL)
# Reset IDS event counts periodically to avoid stale data accumulation
    def periodic_ids_reset(self):
        while self.running:
            time.sleep(CONFIG_FIREWALL_CHECK_INTERVAL * 2)
```

```
            self.ids.reset_event_counts()
# Start periodic IDS reset in a separate thread
    def start_ids_reset_thread(self):
        self.ids_reset_thread = threading.Thread(target=self.periodic_ids_reset)
        self.ids_reset_thread.start()
        logging.info("CyberSecuritySystem: Started IDS reset thread")
# Stop periodic IDS reset thread
    def stop_ids_reset_thread(self):
        if hasattr(self, 'ids_reset_thread'):
            self.ids_reset_thread.join()
            logging.info("CyberSecuritySystem: Stopped IDS reset thread")
# Define a function to simulate external attack traffic
def simulate_external_attack(system, duration=30):
    start_time = time.time()
    while time.time() - start_time < duration:
        ip = random.choice(SUSPICIOUS_IPS)
        packet = {
            "src_ip": ip,
            "dst_ip": "10.0.0." + str(random.randint(1, 254)),
            "port": random.randint(1, 65535),
            "payload": "ATTACK" + str(random.randint(1000, 9999))
        }
        system.ids.analyze_packet(packet)
        time.sleep(0.5)
    logging.info("simulate_external_attack: Completed external attack
simulation")
# Define the main function to run the cybersecurity system
def main():
    system = CyberSecuritySystem()
    system.start_system()
    system.start_ids_reset_thread()
    logging.info("main: Cybersecurity system is now running")
    attack_thread = threading.Thread(target=simulate_external_attack,
args=(system, 30))
    attack_thread.start()
    run_duration = 60
    start_time = time.time()
    while time.time() - start_time < run_duration:
        time.sleep(1)
    if attack_thread.is_alive():
        attack_thread.join()
    system.stop_system()
    system.stop_ids_reset_thread()
    alert_log = system.alert_system.get_alert_log()
    logging.info("main: Total alerts generated: " + str(len(alert_log)))
    for alert in alert_log:
        logging.info("main: " + alert)
    logging.info("main: Cybersecurity system shutdown completed")
# Entry point for the program execution
if __name__ == "__main__":
    main()
```

Файл LogStorageAnalysis.py

```
import threading
import time
import sqlite3
import random
import requests
from flask import Flask, jsonify, request, render_template_string
from sklearn.linear_model import LogisticRegression
import numpy as np
import logging
logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s -
%(message)s')
class LogStorageAnalysis:
    def __init__(self, db_path='logs.db'):
        self.db_path = db_path
        self.connection = sqlite3.connect(self.db_path, check_same_thread=False)
        self.cursor = self.connection.cursor()
        self.cursor.execute('CREATE TABLE IF NOT EXISTS logs (id INTEGER PRIMARY
KEY AUTOINCREMENT, timestamp TEXT, level TEXT, message TEXT)')
        self.connection.commit()
    def store_log(self, timestamp, level, message):
        self.cursor.execute('INSERT INTO logs (timestamp, level, message) VALUES
(?, ?, ?)', (timestamp, level, message))
        self.connection.commit()
    def get_all_logs(self):
        self.cursor.execute('SELECT * FROM logs')
        return self.cursor.fetchall()
    def analyze_logs(self):
        self.cursor.execute('SELECT level, COUNT(*) FROM logs GROUP BY level')
        return self.cursor.fetchall()
class AutomatedResponseModule:
    def __init__(self, log_storage):
        self.log_storage = log_storage
        self.incident_history = []
    def respond_to_incident(self, incident):
        response = {'action': 'block_ip', 'ip': incident.get('ip', ''),
'timestamp': time.strftime("%Y-%m-%d %H:%M:%S")}
        self.incident_history.append(response)
        self.log_storage.store_log(response['timestamp'], 'ALERT', 'Automated
response executed: ' + str(response))
        return response
    def escalate_incident(self, incident):
        escalation = {'action': 'escalate', 'incident': incident, 'timestamp':
time.strftime("%Y-%m-%d %H:%M:%S")}
        self.incident_history.append(escalation)
        self.log_storage.store_log(escalation['timestamp'], 'CRITICAL',
'Incident escalated: ' + str(escalation))
        return escalation
    def get_incident_history(self):
        return self.incident_history
class ThreatIntelligenceFeed:
    def __init__(self, feed_url='https://example.com/threatfeed',
update_interval=30):
        self.feed_url = feed_url
        self.update_interval = update_interval
        self.threat_data = []
        self.running = True
        self.thread = threading.Thread(target=self.update_feed)
        self.thread.daemon = True
    def start(self):
        self.thread.start()
    def stop(self):
```

```
        self.running = False
        self.thread.join()
    def update_feed(self):
        while self.running:
            try:
                response = requests.get(self.feed_url, timeout=5)
                if response.status_code == 200:
                    data = response.json()
                    self.threat_data = data.get('malicious_ips', [])
                else:
                    self.threat_data = []
            except Exception as e:
                self.threat_data = []
            time.sleep(self.update_interval)
    def get_threat_data(self):
        return self.threat_data
class AttackPredictionML:
    def __init__(self):
        self.model = LogisticRegression()
        self.trained = False
    def train_model(self):
        X = np.random.rand(100, 5)
        y = np.random.randint(0, 2, 100)
        self.model.fit(X, y)
        self.trained = True
    def predict_attack(self, features):
        if not self.trained:
            self.train_model()
        features = np.array(features).reshape(1, -1)
        prediction = self.model.predict(features)
        probability = self.model.predict_proba(features)
        return prediction[0], probability[0].tolist()
    def simulate_feature_extraction(self):
        features = [random.random() for _ in range(5)]
        return features
class Dashboard:
    def __init__(self, log_storage, threat_feed, response_module):
        self.app = Flask(__name__)
        self.log_storage = log_storage
        self.threat_feed = threat_feed
        self.response_module = response_module
        self.setup_routes()
    def setup_routes(self):
        @self.app.route('/')
        def home():
            template = "<html><head><title>Cyber Security
Dashboard</title></head><body><h1>Dashboard</h1><p><a href='/logs'>View
Logs</a></p><p><a href='/threats'>View Threat Feed</a></p><p><a
href='/incidents'>View Incident History</a></p></body></html>"
            return render_template_string(template)
        @self.app.route('/logs')
        def logs():
            logs = self.log_storage.get_all_logs()
            return jsonify(logs)
        @self.app.route('/threats')
        def threats():
            threats = self.threat_feed.get_threat_data()
            return jsonify(threats)
        @self.app.route('/incidents')
        def incidents():
            incidents = self.response_module.get_incident_history()
            return jsonify(incidents)
    def run(self, host='0.0.0.0', port=5000):
```

```
        self.app.run(host=host, port=port)
def main():
    log_storage = LogStorageAnalysis()
    response_module = AutomatedResponseModule(log_storage)
    threat_feed = ThreatIntelligenceFeed()
    threat_feed.start()
    attack_ml = AttackPredictionML()
    attack_ml.train_model()
    def simulation_loop():
        while True:
            features = attack_ml.simulate_feature_extraction()
            prediction, probability = attack_ml.predict_attack(features)
            if prediction == 1:
                incident = {'ip': '192.168.' + str(random.randint(0,255)) + '.'
+ str(random.randint(0,255)), 'features': features, 'probability': probability}
                response_module.respond_to_incident(incident)
            time.sleep(5)
    sim_thread = threading.Thread(target=simulation_loop)
    sim_thread.daemon = True
    sim_thread.start()
    dashboard = Dashboard(log_storage, threat_feed, response_module)
    dashboard_thread = threading.Thread(target=dashboard.run, kwargs={'host':
'0.0.0.0', 'port': 5000})
    dashboard_thread.daemon = True
    dashboard_thread.start()
    try:
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        threat_feed.stop()
        exit(0)
if __name__ == "__main__":
    main()
```

Файл XUserBehaviorAnalytics.py

```
import random
import time
import threading
import re
import smtplib
import sqlite3
import logging
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s -
%(message)s')
class UserBehaviorAnalytics:
    def __init__(self):
        self.user_events = []
        self.lock = threading.Lock()
    def record_event(self, user_id, event_type, timestamp):
        with self.lock:
            self.user_events.append({'user_id': user_id, 'event_type':
event_type, 'timestamp': timestamp})
    def analyze_behavior(self):
        analysis = {}
        with self.lock:
            for event in self.user_events:
                uid = event['user_id']
                if uid not in analysis:
                    analysis[uid] = {}
                etype = event['event_type']
                analysis[uid][etype] = analysis[uid].get(etype, 0) + 1
        return analysis
    def simulate_user_activity(self):
        user_ids = ['user1', 'user2', 'user3', 'user4', 'user5']
        event_types = ['login', 'file_access', 'logout', 'download', 'upload']
        while True:
            uid = random.choice(user_ids)
            etype = random.choice(event_types)
            ts = time.strftime("%Y-%m-%d %H:%M:%S")
            self.record_event(uid, etype, ts)
            time.sleep(random.uniform(0.1, 1.0))
class IntelligentFirewallRules:
    def __init__(self):
        self.current_rules = {}
        self.lock = threading.Lock()
    def update_rule(self, ip, risk_score):
        with self.lock:
            self.current_rules[ip] = risk_score
    def recalculate_rules(self):
        new_rules = {}
        for i in range(5):
            ip = "192.168.0." + str(random.randint(1, 254))
            risk = random.uniform(0, 1)
            new_rules[ip] = risk
        with self.lock:
            self.current_rules = new_rules
    def get_rules(self):
        with self.lock:
            return dict(self.current_rules)
    def start_rule_update(self):
        def run():
            while True:
                self.recalculate_rules()
                time.sleep(10)
```

```
        t = threading.Thread(target=run)
        t.daemon = True
        t.start()
class DeepPacketInspection:
    def __init__(self):
        self.anomaly_logs = []
        self.lock = threading.Lock()
    def inspect_packet(self, packet):
        if re.search(r"(malware|virus|attack)", packet.get("payload",
"").lower()):
            with self.lock:
                self.anomaly_logs.append(packet)
            return True
        return False
    def get_anomalies(self):
        with self.lock:
            return list(self.anomaly_logs)
    def simulate_packet_inspection(self):
        sample_payloads = ["normal data", "this contains malware", "attack
detected", "safe content", "virus signature found"]
        while True:
            packet = {"src_ip": "10.0.0." + str(random.randint(1,254)),
"dst_ip": "192.168.1." + str(random.randint(1,254)), "payload":
random.choice(sample_payloads)}
            self.inspect_packet(packet)
            time.sleep(random.uniform(0.5, 2.0))
class NotificationSystem:
    def __init__(self, smtp_server='smtp.example.com', smtp_port=587,
smtp_user='user@example.com', smtp_password='password'):
        self.smtp_server = smtp_server
        self.smtp_port = smtp_port
        self.smtp_user = smtp_user
        self.smtp_password = smtp_password
    def send_email(self, to_email, subject, message):
        msg = MIMEMultipart()
        msg['From'] = self.smtp_user
        msg['To'] = to_email
        msg['Subject'] = subject
        msg.attach(MIMEText(message, 'plain'))
        try:
            server = smtplib.SMTP(self.smtp_server, self.smtp_port)
            server.starttls()
            server.login(self.smtp_user, self.smtp_password)
            server.sendmail(self.smtp_user, to_email, msg.as_string())
            server.quit()
        except Exception as e:
            logging.error("Email sending failed: " + str(e))
    def send_sms(self, phone_number, message):
        logging.info("SMS sent to " + phone_number + ": " + message)
    def notify(self, recipient_email, phone_number, subject, message):
        self.send_email(recipient_email, subject, message)
        self.send_sms(phone_number, message)
class InternalThreatSimulator:
    def __init__(self):
        self.threat_logs = []
        self.lock = threading.Lock()
    def simulate_threat(self):
        threat_types = ['privilege_escalation', 'data_exfiltration',
'unauthorized_access', 'insider_attack']
        threat = random.choice(threat_types)
        user = 'user' + str(random.randint(1, 5))
        timestamp = time.strftime("%Y-%m-%d %H:%M:%S")
        record = {'user': user, 'threat': threat, 'timestamp': timestamp}
```

```
        with self.lock:
            self.threat_logs.append(record)
        return record
    def start_simulation(self):
        def run():
            while True:
                self.simulate_threat()
                time.sleep(random.uniform(2, 5))
        t = threading.Thread(target=run)
        t.daemon = True
        t.start()
    def get_threat_logs(self):
        with self.lock:
            return list(self.threat_logs)
def main():
    uba = UserBehaviorAnalytics()
    t1 = threading.Thread(target=uba.simulate_user_activity)
    t1.daemon = True
    t1.start()
    ifr = IntelligentFirewallRules()
    ifr.start_rule_update()
    dpi = DeepPacketInspection()
    t2 = threading.Thread(target=dpi.simulate_packet_inspection)
    t2.daemon = True
    t2.start()
    notifier = NotificationSystem()
    its = InternalThreatSimulator()
    its.start_simulation()
    def periodic_analysis():
        while True:
            behavior = uba.analyze_behavior()
            rules = ifr.get_rules()
            anomalies = dpi.get_anomalies()
            threats = its.get_threat_logs()
            subject = "Cybersecurity Alert"
            message = "User Behavior: " + str(behavior) + "\nFirewall Rules: " +
str(rules) + "\nAnomalies: " + str(anomalies) + "\nInternal Threats: " +
str(threats)
            notifier.notify("admin@example.com", "+1234567890", subject,
message)
            time.sleep(15)
    t3 = threading.Thread(target=periodic_analysis)
    t3.daemon = True
    t3.start()
    while True:
        time.sleep(1)
if __name__ == "__main__":
    main()
```

Файл DataEncryptionSupport.py

```
import threading
import time
import random
import sqlite3
import json
import os
import logging
from flask import Flask, request, jsonify, session, redirect, url_for,
render_template_string
from werkzeug.security import generate_password_hash, check_password_hash
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s -
%(message)s')
class SIEMIntegration:
    def __init__(self, db_path='siem_logs.db'):
        self.db_path = db_path
        self.connection = sqlite3.connect(self.db_path, check_same_thread=False)
        self.cursor = self.connection.cursor()
        self.cursor.execute('CREATE TABLE IF NOT EXISTS logs (id INTEGER PRIMARY
KEY AUTOINCREMENT, timestamp TEXT, source TEXT, event TEXT, ip TEXT)')
        self.connection.commit()
    def add_log(self, timestamp, source, event, ip):
        self.cursor.execute('INSERT INTO logs (timestamp, source, event, ip)
VALUES (?, ?, ?, ?)', (timestamp, source, event, ip))
        self.connection.commit()
    def correlate_events(self):
        self.cursor.execute('SELECT ip, COUNT(*) FROM logs GROUP BY ip')
        return self.cursor.fetchall()
    def generate_report(self):
        events = self.correlate_events()
        report = "SIEM Report:\n"
        for ip, count in events:
            report += "IP: " + ip + " - Events: " + str(count) + "\n"
        return report
class BotnetDetection:
    def __init__(self, threshold=10):
        self.ip_counts = {}
        self.threshold = threshold
        self.suspicious_ips = set()
        self.lock = threading.Lock()
    def process_packet(self, packet):
        ip = packet.get("src_ip", "")
        with self.lock:
            self.ip_counts[ip] = self.ip_counts.get(ip, 0) + 1
            if self.ip_counts[ip] >= self.threshold:
                self.suspicious_ips.add(ip)
    def simulate_traffic(self):
        while True:
            packet = {"src_ip": "192.168.1." + str(random.randint(1,254)),
"dst_ip": "10.0.0." + str(random.randint(1,254)), "payload": "packet data " +
str(random.randint(1000,9999))}
            self.process_packet(packet)
            time.sleep(0.1)
    def get_report(self):
        with self.lock:
            return list(self.suspicious_ips)
class DataEncryptionSupport:
    def __init__(self):
        self.key = get_random_bytes(16)
    def pad(self, data):
```

```
        pad_len = 16 - (len(data) % 16)
        return data + chr(pad_len) * pad_len
    def unpad(self, data):
        pad_len = ord(data[-1])
        return data[:-pad_len]
    def encrypt_data(self, plaintext):
        cipher = AES.new(self.key, AES.MODE_CBC)
        padded = self.pad(plaintext)
        ciphertext = cipher.encrypt(padded.encode('utf-8'))
        return cipher.iv + ciphertext
    def decrypt_data(self, ciphertext):
        iv = ciphertext[:16]
        actual_ciphertext = ciphertext[16:]
        cipher = AES.new(self.key, AES.MODE_CBC, iv)
        padded = cipher.decrypt(actual_ciphertext).decode('utf-8')
        return self.unpad(padded)
class RecoverySystem:
    def __init__(self, backup_file='system_backup.json'):
        self.state = {"firewall_rules": {}, "user_sessions": {}, "logs": []}
        self.backup_file = backup_file
        self.lock = threading.Lock()
    def update_state(self, key, value):
        with self.lock:
            self.state[key] = value
    def add_log(self, log):
        with self.lock:
            self.state["logs"].append(log)
    def backup_state(self):
        with self.lock:
            with open(self.backup_file, 'w') as f:
                json.dump(self.state, f)
    def restore_state(self):
        if os.path.exists(self.backup_file):
            with open(self.backup_file, 'r') as f:
                with self.lock:
                    self.state = json.load(f)
    def simulate_attack(self):
        with self.lock:
            self.state["firewall_rules"] = {}
            self.state["user_sessions"] = {}
            self.state["logs"].append("Attack occurred at " + time.strftime("%Y-
%m-%d %H:%M:%S"))
    def rollback(self):
        self.restore_state()
class MultiUserManagement:
    def __init__(self, db_path='users.db', secret_key='secret'):
        self.app = Flask(__name__)
        self.app.config['SECRET_KEY'] = secret_key
        self.db_path = db_path
        self.init_db()
        self.setup_routes()
    def init_db(self):
        conn = sqlite3.connect(self.db_path)
        c = conn.cursor()
        c.execute('CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY
AUTOINCREMENT, username TEXT UNIQUE, password TEXT, role TEXT)')
        conn.commit()
        conn.close()
    def register_user(self, username, password, role):
        conn = sqlite3.connect(self.db_path)
        c = conn.cursor()
        try:
```

```
            c.execute('INSERT INTO users (username, password, role) VALUES (?,
?, ?)', (username, generate_password_hash(password), role))
            conn.commit()
        except Exception as e:
            conn.close()
            return False
        conn.close()
        return True
    def authenticate_user(self, username, password):
        conn = sqlite3.connect(self.db_path)
        c = conn.cursor()
        c.execute('SELECT password FROM users WHERE username=?', (username,))
        row = c.fetchone()
        conn.close()
        if row and check_password_hash(row[0], password):
            return True
        return False
    def setup_routes(self):
        @self.app.route('/register', methods=['GET', 'POST'])
        def register():
            if request.method == 'POST':
                username = request.form.get('username')
                password = request.form.get('password')
                role = request.form.get('role', 'user')
                if self.register_user(username, password, role):
                    return redirect(url_for('login'))
                return "Registration Failed"
            return render_template_string("<form method='post'><input
name='username' placeholder='Username'><input name='password' type='password'
placeholder='Password'><input name='role' placeholder='Role'><input
type='submit' value='Register'></form>")
        @self.app.route('/login', methods=['GET', 'POST'])
        def login():
            if request.method == 'POST':
                username = request.form.get('username')
                password = request.form.get('password')
                if self.authenticate_user(username, password):
                    session['username'] = username
                    return redirect(url_for('dashboard'))
                return "Login Failed"
            return render_template_string("<form method='post'><input
name='username' placeholder='Username'><input name='password' type='password'
placeholder='Password'><input type='submit' value='Login'></form>")
        @self.app.route('/logout')
        def logout():
            session.pop('username', None)
            return redirect(url_for('login'))
        @self.app.route('/dashboard')
        def dashboard():
            if 'username' in session:
                username = session['username']
                return "Welcome " + username
            return redirect(url_for('login'))
    def run(self, host='0.0.0.0', port=6000):
        self.app.run(host=host, port=port)
def main():
    siem = SIEMIntegration()
    def siem_simulation():
        while True:
            timestamp = time.strftime("%Y-%m-%d %H:%M:%S")
            source = random.choice(["firewall", "ids", "vpn", "endpoint"])
            event = random.choice(["login attempt", "port scan", "malware
detected", "anomaly detected"])
```

```
            ip = "10.0.0." + str(random.randint(1,254))
            siem.add_log(timestamp, source, event, ip)
            time.sleep(0.2)
    threading.Thread(target=siem_simulation, daemon=True).start()
    botnet = BotnetDetection()
    threading.Thread(target=botnet.simulate_traffic, daemon=True).start()
    encryption = DataEncryptionSupport()
    def encryption_simulation():
        while True:
            data = "Sensitive log at " + time.strftime("%Y-%m-%d %H:%M:%S")
            encrypted = encryption.encrypt_data(data)
            decrypted = encryption.decrypt_data(encrypted)
            time.sleep(0.3)
    threading.Thread(target=encryption_simulation, daemon=True).start()
    recovery = RecoverySystem()
    def recovery_simulation():
        while True:
            recovery.backup_state()
            time.sleep(5)
            recovery.simulate_attack()
            time.sleep(1)
            recovery.rollback()
            time.sleep(5)
    threading.Thread(target=recovery_simulation, daemon=True).start()
    user_management = MultiUserManagement()
    threading.Thread(target=user_management.run, kwargs={'host': '0.0.0.0',
'port': 6000}, daemon=True).start()
    while True:
        report = siem.generate_report()
        botnet_report = botnet.get_report()
        logging.info(report)
        logging.info("Botnet Suspicious IPs: " + str(botnet_report))
        time.sleep(10)
if __name__ == "__main__":
    main()
```

Файл transmitter.py

```
import time
import json

from ..helpers import getLogger, logging
from .. import version

# Gets the instance of the logger.
logSys = getLogger(__name__)


class Transmitter:

	##
	# Constructor.
	#
	# @param The server reference

	def __init__(self, server):
		self.__server = server
		self.__quiet = 0

	##
	# Proceeds a command.
	#
	# Proceeds an incoming command.
	# @param command The incoming command

	def proceed(self, command):
		# Deserialize object
		logSys.log(5, "Command: %r", command)
		try:
			ret = self.__commandHandler(command)
			ack = 0, ret
		except Exception as e:
			logSys.error("Command %r has failed. Received %r",
                                     command, e,

	exc_info=logSys.getEffectiveLevel()<=logging.DEBUG)
			ack = 1, e
		return ack

	##
	# Handle an command.
	#
	#

	def __commandHandler(self, command):
		name = command[0]
		if name == "ping":
			return "pong"
		elif name == "add":
			name = command[1]
			if name == "--all":
				raise Exception("Reserved name %r" % (name,))
			try:
				backend = command[2]
			except IndexError:
				backend = "auto"
			self.__server.addJail(name, backend)
			return name
		elif name == "multi-set":
			return self.__commandSet(command[1:], True)
```

```
elif name == "set":
	return self.__commandSet(command[1:])
elif name == "start":
	name = command[1]
	self.__server.startJail(name)
	return None
elif name == "stop":
	if len(command) == 1:
		self.__server.quit()
	elif command[1] == "--all":
		self.__server.stopAllJail()
	else:
		name = command[1]
		self.__server.stopJail(name)
	return None
elif name == "reload":
	opts = command[1:3]
	self.__quiet = 1
	try:
		self.__server.reloadJails(*opts, begin=True)
		for cmd in command[3]:
			self.__commandHandler(cmd)
	finally:
		self.__quiet = 0
		self.__server.reloadJails(*opts, begin=False)
	return 'OK'
elif name == "unban" and len(command) >= 2:
	# unban in all jails:
	value = command[1:]
	# if all ips:
	if len(value) == 1 and value[0] == "--all":
		return self.__server.setUnbanIP()
	return self.__server.setUnbanIP(None, value)
elif name == "banned":
	# check IP is banned in all jails:
	return self.__server.banned(None, command[1:])
elif name == "echo":
	return command[1:]
elif name == "server-status":
	logSys.debug("Status: ready")
	return "Server ready"
elif name == "server-stream":
	self.__quiet = 1
	try:
		for cmd in command[1]:
			self.__commandHandler(cmd)
	finally:
		self.__quiet = 0
	return None
elif name == "sleep":
	value = command[1]
	time.sleep(float(value))
	return None
elif name == "flushlogs":
	return self.__server.flushLogs()
elif name == "get":
	return self.__commandGet(command[1:])
elif name == "status":
	return self.status(command[1:])
elif name in ("stats", "statistic", "statistics"):
	return self.__server.status("--all", "stats")
elif name == "version":
	return version.version
```

```
		elif name == "config-error":
			logSys.error(command[1])
			return None
		raise Exception("Invalid command")

	def __commandSet(self, command, multiple=False):
		name = command[0]
		# Logging
		if name == "loglevel":
			value = command[1]
			self.__server.setLogLevel(value)
			if self.__quiet: return
			return self.__server.getLogLevel()
		elif name == "logtarget":
			value = command[1]
			if self.__server.setLogTarget(value):
				if self.__quiet: return
				return self.__server.getLogTarget()
			else:
				raise Exception("Failed to change log target")
		elif name == "syslogsocket":
			value = command[1]
			if self.__server.setSyslogSocket(value):
				if self.__quiet: return
				return self.__server.getSyslogSocket()
			else:
				raise Exception("Failed to change syslog socket")
		elif name == "allowipv6":
			value = command[1]
			self.__server.setIPv6IsAllowed(value)
			if self.__quiet: return
			return value
		#Thread
		elif name == "thread":
			value = command[1]
			return self.__server.setThreadOptions(value)
		#Database
		elif name == "dbfile":
			self.__server.setDatabase(command[1])
			db = self.__server.getDatabase()
			if db is None:
				return None
			else:
				if self.__quiet: return
				return db.filename
		elif name == "dbmaxmatches":
			db = self.__server.getDatabase()
			if db is None:
				logSys.log(logging.MSG, "dbmaxmatches setting was not in
effect since no db yet")
				return None
			else:
				db.maxMatches = int(command[1])
				if self.__quiet: return
				return db.maxMatches
		elif name == "dbpurgeage":
			db = self.__server.getDatabase()
			if db is None:
				logSys.log(logging.MSG, "dbpurgeage setting was not in
effect since no db yet")
				return None
			else:
				db.purgeage = command[1]
```

```
                        if self.__quiet: return
                        return db.purgeage
                # Jail
                elif command[1] == "idle":
                        if command[2] == "on":
                                self.__server.setIdleJail(name, True)
                        elif command[2] == "off":
                                self.__server.setIdleJail(name, False)
                        else:
                                raise Exception("Invalid idle option, must be 'on' or
'off'")
                        if self.__quiet: return
                        return self.__server.getIdleJail(name)
                # Filter
                elif command[1] == "ignoreself":
                        value = command[2]
                        self.__server.setIgnoreSelf(name, value)
                        if self.__quiet: return
                        return self.__server.getIgnoreSelf(name)
                elif command[1] == "addignoreip":
                        for value in command[2:]:
                                self.__server.addIgnoreIP(name, value)
                        if self.__quiet: return
                        return self.__server.getIgnoreIP(name)
                elif command[1] == "delignoreip":
                        value = command[2]
                        self.__server.delIgnoreIP(name, value)
                        if self.__quiet: return
                        return self.__server.getIgnoreIP(name)
                elif command[1] == "ignorecommand":
                        value = command[2]
                        self.__server.setIgnoreCommand(name, value)
                        if self.__quiet: return
                        return self.__server.getIgnoreCommand(name)
                elif command[1] == "ignorecache":
                        value = command[2]
                        self.__server.setIgnoreCache(name, value)
                        if self.__quiet: return
                        return self.__server.getIgnoreCache(name)
                elif command[1] == "addlogpath":
                        value = command[2]
                        tail = False
                        if len(command) == 4:
                                if command[3].lower()  == "tail":
                                        tail = True
                                elif command[3].lower() != "head":
                                        raise ValueError("File option must be 'head' or
'tail'")
                        elif len(command) > 4:
                                raise ValueError("Only one file can be added at a time")
                        self.__server.addLogPath(name, value, tail)
                        if self.__quiet: return
                        return self.__server.getLogPath(name)
                elif command[1] == "dellogpath":
                        value = command[2]
                        self.__server.delLogPath(name, value)
                        if self.__quiet: return
                        return self.__server.getLogPath(name)
                elif command[1] == "logencoding":
                        value = command[2]
                        self.__server.setLogEncoding(name, value)
                        if self.__quiet: return
                        return self.__server.getLogEncoding(name)
```

```
elif command[1] == "addjournalmatch": # pragma: systemd no cover
      value = command[2:]
      self.__server.addJournalMatch(name, value)
      if self.__quiet: return
      return self.__server.getJournalMatch(name)
elif command[1] == "deljournalmatch": # pragma: systemd no cover
      value = command[2:]
      self.__server.delJournalMatch(name, value)
      if self.__quiet: return
      return self.__server.getJournalMatch(name)
elif command[1] == "prefregex":
      value = command[2]
      self.__server.setPrefRegex(name, value)
      if self.__quiet: return
      v = self.__server.getPrefRegex(name)
      return v.getRegex() if v else ""
elif command[1] == "addfailregex":
      value = command[2]
      self.__server.addFailRegex(name, value, multiple=multiple)
      if multiple:
            return True
      if self.__quiet: return
      return self.__server.getFailRegex(name)
elif command[1] == "delfailregex":
      value = int(command[2])
      self.__server.delFailRegex(name, value)
      if self.__quiet: return
      return self.__server.getFailRegex(name)
elif command[1] == "addignoreregex":
      value = command[2]
      self.__server.addIgnoreRegex(name, value, multiple=multiple)
      if multiple:
            return True
      if self.__quiet: return
      return self.__server.getIgnoreRegex(name)
elif command[1] == "delignoreregex":
      value = int(command[2])
      self.__server.delIgnoreRegex(name, value)
      if self.__quiet: return
      return self.__server.getIgnoreRegex(name)
elif command[1] == "usedns":
      value = command[2]
      self.__server.setUseDns(name, value)
      if self.__quiet: return
      return self.__server.getUseDns(name)
elif command[1] == "findtime":
      value = command[2]
      self.__server.setFindTime(name, value)
      if self.__quiet: return
      return self.__server.getFindTime(name)
elif command[1] == "datepattern":
      value = command[2]
      self.__server.setDatePattern(name, value)
      if self.__quiet: return
      return self.__server.getDatePattern(name)
elif command[1] == "logtimezone":
      value = command[2]
      self.__server.setLogTimeZone(name, value)
      if self.__quiet: return
      return self.__server.getLogTimeZone(name)
elif command[1] == "maxmatches":
      value = command[2]
      self.__server.setMaxMatches(name, int(value))
```

```
                if self.__quiet: return
                return self.__server.getMaxMatches(name)
            elif command[1] == "maxretry":
                value = command[2]
                self.__server.setMaxRetry(name, int(value))
                if self.__quiet: return
                return self.__server.getMaxRetry(name)
            elif command[1] == "maxlines":
                value = command[2]
                self.__server.setMaxLines(name, int(value))
                if self.__quiet: return
                return self.__server.getMaxLines(name)
            # command
            elif command[1] == "bantime":
                value = command[2]
                self.__server.setBanTime(name, value)
                if self.__quiet: return
                return self.__server.getBanTime(name)
            elif command[1] == "attempt":
                value = command[2:]
                if self.__quiet: return
                return self.__server.addAttemptIP(name, *value)
            elif command[1].startswith("bantime."):
                value = command[2]
                opt = command[1][len("bantime."):]
                self.__server.setBanTimeExtra(name, opt, value)
                if self.__quiet: return
                return self.__server.getBanTimeExtra(name, opt)
            elif command[1] == "banip":
                value = command[2:]
                return self.__server.setBanIP(name,value)
            elif command[1] == "unbanip":
                ifexists = True
                if command[2] != "--report-absent":
                    value = command[2:]
                else:
                    ifexists = False
                    value = command[3:]
                return self.__server.setUnbanIP(name, value,
ifexists=ifexists)
            elif command[1] == "addaction":
                args = [command[2]]
                if len(command) > 3:
                    args.extend([command[3], json.loads(command[4])])
                self.__server.addAction(name, *args)
                if self.__quiet: return
                return args[0]
            elif command[1] == "delaction":
                value = command[2]
                self.__server.delAction(name, value)
                return None
            elif command[1] == "action":
                actionname = command[2]
                action = self.__server.getAction(name, actionname)
                if multiple:
                    for cmd in command[3]:
                        logSys.log(5, "  %r", cmd)
                        actionkey = cmd[0]
                        if callable(getattr(action, actionkey, None)):
                            actionvalue = json.loads(cmd[1]) if
len(cmd)>1 else {}
                            getattr(action, actionkey)(**actionvalue)
                        else:
```

```
                                    actionvalue = cmd[1]
                                    setattr(action, actionkey, actionvalue)
                        return True
                  else:
                        actionkey = command[3]
                        if callable(getattr(action, actionkey, None)):
                              actionvalue = json.loads(command[4]) if
len(command)>4 else {}
                              if self.__quiet: return
                              return getattr(action, actionkey)(**actionvalue)
                        else:
                              actionvalue = command[4]
                              setattr(action, actionkey, actionvalue)
                              if self.__quiet: return
                              return getattr(action, actionkey)
            raise Exception("Invalid command %r (no set action or not yet
implemented)" % (command[1],))

      def __commandGet(self, command):
            name = command[0]
            # Logging
            if name == "loglevel":
                  return self.__server.getLogLevel()
            elif name == "logtarget":
                  return self.__server.getLogTarget()
            elif name == "syslogsocket":
                  return self.__server.getSyslogSocket()
            #Thread
            elif name == "thread":
                  return self.__server.getThreadOptions()
            #Database
            elif name == "dbfile":
                  db = self.__server.getDatabase()
                  if db is None:
                        return None
                  else:
                        return db.filename
            elif name == "dbmaxmatches":
                  db = self.__server.getDatabase()
                  if db is None:
                        return None
                  else:
                        return db.maxMatches
            elif name == "dbpurgeage":
                  db = self.__server.getDatabase()
                  if db is None:
                        return None
                  else:
                        return db.purgeage
            # Jail, Filter
            elif command[1] == "banned":
                  # check IP is banned in all jails:
                  return self.__server.banned(name, command[2:])
            elif command[1] == "logpath":
                  return self.__server.getLogPath(name)
            elif command[1] == "logencoding":
                  return self.__server.getLogEncoding(name)
            elif command[1] == "journalmatch": # pragma: systemd no cover
                  return self.__server.getJournalMatch(name)
            elif command[1] == "ignoreself":
                  return self.__server.getIgnoreSelf(name)
            elif command[1] == "ignoreip":
                  return self.__server.getIgnoreIP(name)
```

```
            elif command[1] == "ignorecommand":
                  return self.__server.getIgnoreCommand(name)
            elif command[1] == "ignorecache":
                  return self.__server.getIgnoreCache(name)
            elif command[1] == "prefregex":
                  v = self.__server.getPrefRegex(name)
                  return v.getRegex() if v else ""
            elif command[1] == "failregex":
                  return self.__server.getFailRegex(name)
            elif command[1] == "ignoreregex":
                  return self.__server.getIgnoreRegex(name)
            elif command[1] == "usedns":
                  return self.__server.getUseDns(name)
            elif command[1] == "findtime":
                  return self.__server.getFindTime(name)
            elif command[1] == "datepattern":
                  return self.__server.getDatePattern(name)
            elif command[1] == "logtimezone":
                  return self.__server.getLogTimeZone(name)
            elif command[1] == "maxmatches":
                  return self.__server.getMaxMatches(name)
            elif command[1] == "maxretry":
                  return self.__server.getMaxRetry(name)
            elif command[1] == "maxlines":
                  return self.__server.getMaxLines(name)
            # Action
            elif command[1] == "bantime":
                  return self.__server.getBanTime(name)
            elif command[1] == "banip":
                  return self.__server.getBanList(name,
                        withTime=len(command) > 2 and command[2] == "--with-
time")
            elif command[1].startswith("bantime."):
                  opt = command[1][len("bantime."):]
                  return self.__server.getBanTimeExtra(name, opt)
            elif command[1] == "actions":
                  return list(self.__server.getActions(name).keys())
            elif command[1] == "action":
                  actionname = command[2]
                  actionvalue = command[3]
                  action = self.__server.getAction(name, actionname)
                  return getattr(action, actionvalue)
            elif command[1] == "actionproperties":
                  actionname = command[2]
                  action = self.__server.getAction(name, actionname)
                  return [
                        key for key in dir(action)
                        if not key.startswith("_") and
                              not callable(getattr(action, key))]
            elif command[1] == "actionmethods":
                  actionname = command[2]
                  action = self.__server.getAction(name, actionname)
                  return [
                        key for key in dir(action)
                        if not key.startswith("_") and callable(getattr(action,
key))]
            raise Exception("Invalid command (no get action or not yet
implemented)")

      def status(self, command):
            if len(command) == 0:
                  return self.__server.status()
            elif len(command) >= 1 and len(command) <= 2:
```

```
            name = command[0]
            flavor = command[1] if len(command) == 2 else "basic"
            if name == "--all":
                  return self.__server.status("--all", flavor)
            return self.__server.statusJail(name, flavor=flavor)
      raise Exception("Invalid command (no status)")
```