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 другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи виявлення уразливих додатків у мережевих Cloud-сервісах”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-24М
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Воропай І.В.
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Смірнов С.А.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

АНОТАЦІЯ

Воропай І.В. Дослідження та програмна реалізація системи виявлення уразливих додатків у мережевих Cloud-сервісах. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи виявлення уразливих додатків у мережевих Cloud-сервісах.

Метою розробки є дослідження та програмна реалізація системи виявлення уразливих додатків у мережевих Cloud-сервісах.

Об'єктом дослідження є процес виявлення уразливих додатків у мережевих Cloud-сервісах.

Предметом дослідження є методи виявлення уразливих додатків у мережевих Cloud-сервісах.

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

Результат роботи – програмна реалізація системи виявлення уразливих додатків у мережевих Cloud-сервісах.

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: комп'ютерна інженерія, виявлення уразливих додатків, мережеві Cloud-сервіси

ABSTRACT

Voropai I.V. Research and software implementation of a system for detecting vulnerable applications in network Cloud services. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the second (master's) level of higher education, software has been developed, which is intended for a system for detecting vulnerable applications in network Cloud services.

The purpose of the development is the research and software implementation of a system for detecting vulnerable applications in network Cloud services.

The object of the research is the process of detecting vulnerable applications in network Cloud services.

The subject of the research is methods for detecting vulnerable applications in network Cloud services.

The research methods are based on methods of information protection in the network, methods of mathematical statistics, and methods of software development.

The result of the work is a software implementation of a system for detecting vulnerable applications in network Cloud services.

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 the software are provided.

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

The program was developed in the Python environment.

Keywords: computer engineering, detection of vulnerable applications, network Cloud services

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 3
ВСТУП..... 4
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 7
1.1 Призначення системи..... 7
1.2 Область застосування..... 8
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 10
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти..... 10
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування..... 19
2.3 Розгорнута постановка завдання 19
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 21
3.1 Опис функціонування системи 21
3.2 Розробка структурної схеми..... 30
3.3 Розробка функціональної схеми 35
3.4 Розробка діаграми процесів..... 52
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ..... 54
4.1 Розробка блок-схем та опис алгоритмів функціонування системи..... 54
4.2 Захист розробленого програмного забезпечення..... 68
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 70
6 НАУКОВА НОВИЗНА 75

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ ІТ-ПРОЄКТУ 76
7.1 Визначення цільової аудиторії кінцевого готового продукту 76
7.2 Оцінка привабливості шляхом застосування методів експертних оцінок ... 77
7.3 Вибір методу оцінки вартості ПЗ 77
7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості 78
7.5 Пропозиція алгоритму просування проєкту розробки ПЗ 79
7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ 79
7.7 Визначення ключових факторів успіху конкретного проєкту 80
8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ 81
8.1 Вступ 81
8.2 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста ... 82
8.3 Розробка заходів з умов поліпшення охорони праці 85
8.4 Розрахункова частина 86
9 ОСНОВНІ ВИСНОВКИ 89
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 91

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ДВЧ	–	датчик випадкових чисел
ДСТ 28147-89	–	алгоритм шифрування
ЕОМ	–	електронна обчислювальна машина
ІС	–	інформаційна система
ОС	–	обчислювальна система
ПВЧ	–	псевдовипадкові числа
ПЗ	–	програмне забезпечення
РПЗ	–	руйнуючі програмні засоби
СЗІ	–	система захисту інформації
ASCII	–	система кодування
DES	–	алгоритм шифрування
FEAL	–	алгоритм шифрування
IDEA	–	алгоритм шифрування
KOI-8	–	система кодування
RISC	–	архітектура процесора

ВСТУП

Актуальність теми. При розміщенні клієнтських додатків у хмарі необхідно передбачити міри захисту на випадок наявності в них уразливостей і інших недекларованих можливостей. Забезпечення безпеки хмарного сервісу – досить складне завдання. Зобов'язання щодо доступності сервісу й захищеності даних у ньому – невід'ємна частина пропозиції й важливий пункт SLA. Провайдери хмарних сервісів не скупляться на реалізацію засобів і мер інформаційної безпеки, оскільки захищеність забезпечує ним вагому конкурентну перевагу. Для цього використовуються антивірусні засоби, міжмережні екрани різних рівнів, системи протидії DDoS і запобігання вторгнень, різноманітні «пісочниці», SOC/SIEM і т.п. У гарному хмарному центрі є цілодобова служба моніторингу й відбиття атак, що складає із кваліфікованих фахівців. Однак забезпечення захисту стає куди більше складним завданням, коли хмарний провайдер надає своїм користувачам можливість розміщати їхні власні сервіси по моделі IaaS і PaaS. Тоді в системі, яка захищається, виникає шар клієнтських додатків, контролювати якість яких провайдер не в змозі. Уразливий додаток може стати проблемою не тільки для тих, хто його написав, але й для інших клієнтів хмарного провайдеру. Неодноразово вже виникали ситуації, коли атаці піддавався додаток, розміщений в центрі обробки даних, але під її впливом виявлялися недоступні всі додатки. Крім того, зламавши один додаток, наприклад одержавши до нього привілейований доступ, зловмисник здатний заразити й скомпрометувати інші в цій же хмарі. Як би не були ізольовані ресурси, керування ними (або їхньою частиною) нерідко здійснюється з одного центра.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи виявлення уразливих додатків у мережевих Cloud-сервісах.

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

– Огляд існуючих систем виявлення уразливих додатків у мережевих Cloud-сервісах.

– Дослідження системи виявлення уразливих додатків у мережевих Cloud-сервісах.

– Програмна реалізація системи виявлення уразливих додатків у мережевих Cloud-сервісах.

Об'єктом дослідження є процес виявлення уразливих додатків у мережевих Cloud-сервісах.

Предметом дослідження є методи виявлення уразливих додатків у мережевих Cloud-сервісах.

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

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

– Удосконалено метод виявлення уразливих додатків у мережевих Cloud-сервісах.

– Розроблено вітчизняний продукт виявлення уразливих додатків у мережевих Cloud-сервісах, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі виявлення уразливих додатків у мережевих Cloud-сервісах.

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

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти LV науково-технічної конференції «Наука в ЦНТУ: основні досягнення та перспективи розвитку» (2025 р.), основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №15.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи виявлення уразливих додатків у мережевих Cloud-сервісах, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Надаючи клієнтам API або просто ресурси для хмарного хостингу, його власники повинні бути впевнені в тім, що розміщений у хмарі код не знизить загальну захищеність. Поки ж у більшості хмарних сервісів клієнтські додатки приймаються «як є», а кількість атак на уразливі додатки тільки росте. Як розв'язати цю проблему? Проводити пентести? Організовувати незалежне приймання додатків, як це робиться при випуску тиражируємих продуктів?

Але чи погодяться клієнти хмарних сервісів на такі міри? Чи будуть вони оплачувати дослідження своїх додатків, розміщених у хмарному сервісі? Навряд чи. Здебільшого це бізнесмени, і швидкість запуску сервісів для них важливіше, ніж їхня захищеність. Ресурси, у тому числі людські, у них теж обмежені, і вибір між «додати нові функції» і «виправляти уразливості в старих» для них теж очевидний – не виправляти. Їм немає справи до ризиків інших клієнтів, як і до ризиків, що виходять від інших клієнтів сервісу, – вони підписали SLA з конкретним сервісом і очікують, що всі проблеми із захищеністю будуть вирішуватися провайдером.

Швидше за все, оплачувати такі процедури прийде власникові хмарного сервісу, оскільки уразливість розміщених програм прямо впливає на захищеність сервісу й дотримання SLA. Деякі центри обробки даних, де розміщаються клієнтські додатки, уже вводять ті або інші елементи процесу приймання. Думаю, не за горами й повноцінними процедурами, знайомі по прийманню у фонди алгоритмів і програм.

1.2 Область застосування

Із чого почати? У першу чергу з формулювання вимог для додатків, які будуть розміщатися в хмарному сервісі. Варіанти можуть бути різні – від загальних вимог до стандартів безпечного програмування до приватних вимог щодо використання конкретного хмарного API або специфіки інфраструктури. Мало описати вимоги, потрібно мати можливість їх проконтролювати. Тому провайдер повинен мати продукти або сервіси, призначені для перевірки саме цих вимог, а також персонал, навчений користуватися такими інструментами й інтерпретувати результати їхньої роботи.

Наявності тільки вимог і способів їхнього контролю теж недостатньо для рішення проблеми неякісних додатків. Останнім і самим важким етапом впровадження твердих правил є реалізація зворотного зв'язка: «Ми встановили, що якість продукту, що відправляється в хмару, неприйнятно. Що далі?» Самим складним для реалізації цей елемент є тому, що при його впровадженні виникає конфлікт інтересів бізнесу («Клієнт хоче розмістити свій продукт у нашім хмарному сервісі, давайте скоріше візьмемо з його гроші!») і служб безпеки («Розміщення такого продукту підвищує ризики атак на клієнта, сервіс у цілому й інших клієнтах»). Щораз прийде зіставляти вигоду й ризики.

Тому відносини із клієнтами повинні бути жорстко регламентовані. У соцмережах у користувальницькій угоді передбачається можливість видалення незаконних текстів і навіть відключення (блокування) облікового запису, від імені якої такі тексти розміщаються, – аж до повної заборони. Аналогічно цьому правилу, для хмарного сервісу може бути застережена умова розміщення додатків – відсутність уразливостей і інших недекларованих можливостей, здатних вплинути на доступність сервісу й конфіденційність даних. Зокрема, повинна бути передбачена можливість тимчасової зупинки додатка, якщо в ньому виявлена уразливість, наявність якої може викликати погрози для хмарного сервісу й інших клієнтів. Таку умову варто ввести ще й тому, що за

конфіденційність і доступність даних клієнта відповідає хмарний сервіс і у випадку інциденту складно довести, що клієнт «сам винуватий».

Хмарні сервіси стають усе популярніше, а розташовувані клієнтські додатки – усе складніше. Якщо зараз не задуматися про зростаючі ризики, можна безповоротно упустити час.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи виявлення уразливих додатків у мережевих Cloud-сервісах, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Ми вибрали події першого кварталу 2025 року, які, на наш погляд, ілюструють основні напрямки розвитку в сфері DDoS-атак і інструментів для їхнього здійснення.

Рекордна по потужності DDoS-атака з «відбиттям»

DDoS-атаки, що використовують методи посилення/відбиття, як і раніше не втрачають своєї актуальності й обновляють рекорди пікових потужностей. Методи посилення, з технічної точки зору, не є новим підходом в організації розподілених атак, спрямованих на відмову в обслуговуванні, однак зловмисники знаходять нові можливості й ресурси для нарощування потужності своїх ботнетів. Так, наприклад, відповідно до недавно опублікованого звіту експертів, в 2025 році рекордною стала атака потужністю 450-500 Гбіт/сек.

DDoS-атака на Трампа

Однак не виключено, що рекорд потужності атаки, зафіксований в 2024 році, уже був побитий – і досить швидко. На початку 2025 року офіційний сайт передвиборної кампанії Дональда Трампа піддалися DDoS-атаці, потужність якої, відповідно до неперевірених джерел, склала 602 Гбіт/сек. Відповідальність за те, що відбулося взяла на себе група хакерів New World Hacking.

Використання протоколу DNSSEC

Для здійснення DDoS-атак злочинці всі частіше використовують протокол DNSSEC, завдання якого – мінімізувати атаки, спрямовані на підміну DNS-адреси. Стандартна відповідь по протоколі DNSSEC крім даних про домен містить додаткову автентифікаційну інформацію. Таким чином, на відміну від

стандартної відповіді DNS, що має обсяг 512 байт, відповідь DNSSEC досягає близько 4096 байт. Зловмисники використовують дану особливість для реалізації DDoS-атак методом посилення. Переважно вони використовують домени в урядовій зоні .gov, з тієї причини, що в США такі домени зобов'язані підтримувати DNSSEC за законом.

Pingback-атаки на WordPress

Сайти під керуванням WordPress знову піддаються DDoS-атакам з ботнетів, також побудованих на базі цієї популярної CMS. Для здійснення атак експлуатується функція pingback в CMS WordPress. Вона полягає в повідомленні веб-автора поста, коли хто-небудь в інтернеті посилається на його контент. Якщо адміністратор сайту, що перебуває під керуванням WordPress, включив дану функцію, то всі посилання в опубліковані на цьому сайті матеріалах можуть здійснювати так званий «pingback» – відправляти спеціальний XML-RPC запит на сайт-оригінал. При наявності величезного числа даних pingback-запитів сайт-оригінал може «відмовити в обслуговуванні». Дана функція як і раніше користується увагою з боку зловмисників і допомагає їм реалізувати DDoS-атаки на рівні додатків.

Злом Linux Mint

21 лютого 2025 року глава Linux Mint Клемент Лефевр (Clement Lefebvre) повідомив, що зловмисники зуміли зламати інфраструктуру проекту, включаючи його офіційний сайт і форум, і підмінили посилання на легітимний ISO-образ дистрибутива Linux Mint 17.3 Cinnamon власним URL. Дистрибутив зловмисників містив шкідливий код, що використовує заражені машини для здійснення DDoS-атак.

Атаки на ІБ-компанії

Кіберзлочинці не забувають про компанії, що працюють в області інформаційної безпеки. Всі більш-менш відомі гравці цього ринку, особливо провайдери сервісів анти-DDoS, змушені регулярно відбивати DDoS-атаки,

націлені на їхні ресурси. Помітної шкоди ці акції нанести не можуть, все-таки ці ресурси захищаються дуже якісно, однак зловмисників це не зупиняє.

Як правило, злочинці не ставлять собі завдання «покласти» сайт ІБ-компанії за всяку ціну – атаки тривають недовго, у більшості випадків вони припиняються відразу, як тільки джерело зауважує ознаки роботи захисних систем, адже зловмисники не хочуть впусту виснажувати ботнети, використання яких коштує грошей. І все-таки в довгостроковій перспективі атаки не припиняються.

Найбільш тривала DDoS-атака в першому кварталі 2025 року тривала 197 годин #KLReport

Аналіз переписок на підпільних форумах дозволяє зробити припущення, що в злочинному кіберспівтоваристві прийняте використовувати сайти ІБ-компаній як іспитові мети, для тестування нових методів і інструментів. Підхід не гірше інших, однак, він дає нам у руки коштовні відомості. Якщо статистика по DDoS в усьому світі показує зріз ситуації на теперішній час, атаки на ІБ-компанії дозволяють у якімсь ступені оцінити майбутнє DDoS.

Як і раніше нам доводиться мати справу з атаками з посиленням. Число їх трохи знизилося в порівнянні з минулим роком, зате максимальна потужність зросла вчетверо. Це підтверджує тренд загального посилення таких атак – злочинці змушені нарощувати потужність, щоб перебороти захисні міри з боку інтернет-провайдерів і ІБ-компаній. У нашім випадку жодна з них не привела до перерв у доступності наших сайтів.

Судячи з ряду атак на ресурси у першому кварталі 2025 року, «вершки» кіберпреступников починають згадувати методи, що втратили в останні роки популярність, атакуючи на рівні додатка. Що цікаво, було відзначено кілька атак рівня додатка одночасно на кілька наших ресурсів. Фактично, потужності DDoS-Ресурсів розмивалися на кілька цілей, що знижувало ефект, надаваний на кожну мету. Це можна пояснити тим, що метою зловмисників було не порушення

роботи сайтів, а тестування інструментарію й вивчення відповідної реакції з нашої сторони. Сама тривала атака такого роду тривала менш 6 годин.

В Q1 2025 переважали Windows-ботнети, на них довелося 55,5% від всіх DDoS-атак #KLReport

Ми можемо припустити, що частка атак на канали поступово буде знижуватися, а на передній план вийдуть атаки на додатки й комбіновані атаки (сполучення атак на встаткування й атак на додаток).

Потужні UDP-атаки з посиленням узвичаїлися кілька років назад, і дотепер продовжують залишатися улюбленим інструментом зловмисників. Причини їхньої популярності цілком зрозумілі, тому що атаки цього типу відносно легкі в реалізації, дозволяють забезпечити величезну потужність при відносно невеликому ботнеті, найчастіше задіють третю сторону й украй утрудняють виявлення джерела атаки.

Хоча в першому кварталі 2025 року сервіс DDoS Prevention продовжив відбивати UDP-атаки з посиленням, ми думаємо, що поступово вони будуть сходити зі сцени. Ніколи те, яке здавалось непосильним завдання координації інтернет-провайдерів і ІБ для ефективної фільтрації генеруємого UDP-атаками сміттєвого трафіку вже практично вирішені. Провайдери, що зштовхнулися з реальною погрозою вичерпання магістральних каналів за рахунок значного потоку UDP-пакетів великого обсягу, обзавелися необхідним устаткуванням і компетенціями й «ріжуть» цей трафік на корені. Через цього атаки на канал з посиленням стають усе менш ефективними, а, виходить, на них усе складніше заробляти.

Самими популярними методами DDoS-атак в Q1 2025 залишилися SYN-DDoS, TCP-DDoS і HTTP-DDoS #KLReport.

Атаки рівня додатка на веб-сервіси вимагають для виконання більші ботнети, або кілька високопродуктивних серверів і широкий вихідний канал, а також кропіткої підготовчої роботи з дослідження мети, з'ясуванню її уразливих крапок. Без цього вони неефективні. При цьому грамотно проведену атаку рівня

додатка складно відбити, не заблокувавши доступ легітимним користувачам – шкідливі запити виглядають вірогідно, і кожний бот чесно відпрацьовує процедуру установки з'єднання. Аномальне лише високе навантаження на сервіс. Саме такі спроби ми фіксували в першому кварталі. Це говорить про те, що ринок DDoS розвився настільки, що складні й дорогі у виконанні атаки стають економічно вигідними, і заробляти на цьому намагаються більше кваліфіковані кіберзлочинці.

Більше того, є реальна небезпека масового освоєння цих методів менш кваліфікованими злочинцями – чим вище популярність техніки, тим більше інструментів для її реалізації пропонується на чорному ринку. І якщо атаки рівня додатка дійсно стануть популярними, нам варто очікувати росту числа замовників такого роду DDoS-атак і підвищення кваліфікації виконавців.

Статистика DDoS-атак з використанням ботнетів

Методологія

Система DDoS Intelligence є частиною рішення DDoS Prevention, призначена для перехоплення й аналізу команд, що надходять ботам із серверів керування й контролю, і не вимагає при цьому ні зараження яких-небудь користувальницьких пристроїв, ні реального виконання команд зловмисників.

Даний звіт містить статистику DDoS Intelligence за перший квартал 2025 року.

В Q1 2025 року 93,6% всіх DDoS-атак були націлені на мішені, розташовані в 10 країнах #KLReport.

За окрему (одну) DDoS-атаку в даному звіті приймається та, під час якої перерва між періодами активності ботнета не перевищує 24 годин. Так, наприклад, у випадку, якщо той самий ресурс був атакований тим самим ботнетом з перервою в 24 години й більше, це розглядається як дві атаки. Також за окремі атаки зараховуються запити на один ресурс, але зроблені ботами з різних ботнетів.

Географічне розташування жертв DDoS-атак і серверів, з яких відправлялися команди, визначається по їх IP-адресах. Кількість унікальних мішеней DDoS-атак у даному звіті вважається по числу унікальних IP адрес у квартальній статистиці. Важливо відзначити, що статистика DDoS Intelligence обмежена тільки тими ботнетами, які були виявлені й проаналізовані програмою. Варто також мати що ботнети – це лише один з інструментів здійснення DDoS-атак, і представлені в даному звіті дані не охоплюють усе без винятку DDoS-атаки, що відбулися за зазначений період.

Підсумки кварталу

– У першому кварталі 2025 року DDoS-атакам піддавалися мети, розташовані в 74 країнах миру (в останньому кварталі 2015 року – в 69).

– На ресурси в 10 країнах миру довелося 93,6% відзначених атак.

– І по числу атак, і по кількості цілей DDoS-атак першість залишається за Китаєм, США й Південною Кореєю, однак у цьому кварталі в десятку лідерів потрапили такі європейські країни, як Франція й Німеччина.

– Найбільш тривала DDoS-атака першого кварталу 2025 року тривала 197 годин (8,2 дні), що значно нижче максимуму минулого кварталу (13,9 днів). При цьому почастішали множинні атаки на одну мету (до 33 атак на один ресурс за період).

– Самими популярними методами атак як і раніше залишаються SYN-DDoS, TCP-DDoS і HTTP-DDoS, при цьому із кварталу у квартал відзначається постійне зниження числа UDP-атак.

– В основному командні сервери залишилися в тих же країнах, що й у попередній квартал, але Європа значно додала по цьому показнику – відзначений ріст у Великобританії й Франції.

Географія атак

На початку 2025 року DDoS-атаки були зафіксовані в 74 країнах.

Статистика по кількості атак показує, що 93,6% всіх DDoS-атак доводиться на десять країн.

Розподіл DDoS-атак по країнах, Q1 2025 і Q4 2015

Трійка лідерів за перший квартал 2025 року не змінилася, при цьому значно, з 18,4% до 20,4% підвищилася частка Південної Кореї, і на 2,2 п.п. понизилася частка США. Відзначимо також, що в першому кварталі 2025 року виросла частка атак, націлених на ресурси на Україні – з 0,3% до 2,0%.

Статистика розподілу унікальних мішеней атак свідчить про те, що на мішені в десяти атакованих країнах довелося 94,7% всіх атак.

Розподіл унікальних мішеней DDoS-атак по країнах, Q1 2025 і Q4 2024

На 3,4 п.п. зросло число цілей у Південній Кореї. У той же час показник Китаю понизився з 50,3% у четвертому кварталі 2024 року до 49,7% у першому кварталі 2025 року. Зменшилася також частка мішеней DDoS-атак, які довелися на США (9,6% мішеней у порівнянні з 12,8% у минулому кварталі). При цьому перша трійка країн зберегла свої позиції, з більшим відривом випереджаючи всі інші країни.

В Q1 2025 почастішали множинні атаки на одну мету: до 33 атак на один ресурс #KLReport.

У першому кварталі 2025 року в п'ятірку лідерів по числу DDoS-цілей увійшла Україна, її частка піднялася з незначних 0,5% наприкінці минулого року до 1,9% у першому кварталі 2025 року.

За результатами кварталу з першої десятки атакованих країн вибули Тайвань і Нідерланди, що втратили по 0,8 і 0,7 п.п. відповідно.

Динаміка числа DDoS-атак

У першому кварталі 2025 року відзначений відносно рівний розподіл атак по днях, за винятком різкого зниження 6 лютого. Пікове число атак довелося на 31 березня (1271 атака).

Динаміка числа DDoS-атак*, Q1 2025

У зв'язку з тим, що DDoS-атаки можуть тривати безупинно кілька днів, у таймлайні одна атака може вважатися кілька разів – по одному разі за кожний день.

Як і в минулому кварталі, самим популярним для DDoS день виявився понеділок (16,5% атак), на другому місці – четвер (16,2%). Вівторок, що у минулому кварталі посідав друге місце, здав свої позиції (з 16,4% до 13,4%) і став останнім по DDoS-активності вдень тижня.

Розподіл DDoS-атак по днях тижня

Типи й тривалість DDoS-атак

Рейтинг методів DDoS-атак по популярності майже не міняється із кварталу у квартал. Всі так само найбільше часто застосовуваними залишаються SYN-DDoS, чия частка трохи знизилася в першому кварталі 2025 року (з 57,0% до 54,9%) і TCP-DDoS, що втратив 0,7 п.п. Різко додав ICMP-DDoS, що наростив частку до 9%, але розміщення топ-5 це не змінило.

Розподіл DDoS-атак по типах

Відзначимо, що показники UDP-DDoS знижуються вже протягом року. Із другого кварталу 2015 року вони знизилися майже в 10 разів – з 11,1% до 1,5%.

По тривалості так само, як і в минулому звітному періоді, близько 70% атак доводиться на короткочасні акції тривалістю в 4 години й менш. При цьому істотно знизилася максимальна тривалість атаки. Якщо в останньому кварталі 2024 року була відзначена атака, що тривала 333 години, то в першому кварталі 2025 року сама довга зафіксована нами атака закінчилася вже через 197 годин.

Розподіл DDoS-атак по тривалості, годинники

Командні сервери й типи ботнетів

У першому кварталі 2025 року Південна Корея продовжує беззастережно лідирувати по числу серверів керування й контролю, розташованих на території цієї країни. Її частка ще підвищилася – з 59% за минулий звітний період до 67,6%.

Другий рядок зайняв Китай, частка якого виросла з 8,3% до значних 9,5%. У результаті Китай витиснув США на третє місце (6,8% після 11,5% у четвертому кварталі 2024 року). У топі країн по кількості C&C серверів уперше за

спостережуваний період виявилася Франція, що співвідноситься зі статистикою по кількості, що збільшилася, атак у цій країні.

Розподіл командних серверів ботнетів по країнах, Q1 2025

У першому кварталі 2025 року для атак на 99,73% цілей зловмисники застосовували боти одного із сімейств. Ботами двох різних сімейств (використовуваних одним або декількома виконавцями) протягом кварталу були атаковані 0,25% мішеней. Для атак на 0,01% мішеней проводилися за допомогою ботів трьох різних сімейств. Як і раніше найбільш популярними сімействами ботів залишаються Sotdas, Xor і BillGates.

У першому кварталі 2025 року перевага в співвідношенні атак з Windows- і Linux-ботнетів виявився на стороні Windows-ботнетів. При цьому вужі третій квартал підряд різниця по частках платформ залишається приблизно рівної 10%.

Висновок

Події першого кварталу ще раз продемонстрували той факт, що зловмисники не зупиняються на вже досягнутому й нарощують свої обчислювальні ресурси для здійснення DDoS-атак. Сценарії «посилення», що де-факто стали стандартним інструментом для організації потужної атаки, експлуатують недоліки нових мережних протоколів. При цьому приводи для атак лиходії знаходять зовсім різні: від передвиборних кампаній і атак на ресурси кандидатів, до «розбирань» з іншими учасниками злочинного співтовариства й своїх конкурентів на чорному ринку. Нерідкі випадки, коли під DDoS-атакою виявляються організації, що спеціалізуються на захисті від цих атак. При цьому, з урахуванням поширення уразливих пристроїв і робочих станцій і достатку недоліків конфігурації на рівні додатка, знижується вартість організації більш-менш серйозної атаки. Тому потрібна надійний захист, що у випадку замовленої атаки зробить її фінансово нерентабельною для злочинців.

2.2 Обґрунтування вибору засобів для побудови системи 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 другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи виявлення уразливих додатків у мережевих Cloud-сервісах.

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

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

б) вибрати та обґрунтувати методику побудови системи контролю роботи

технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Загальнодоступні й приватні хмари піддаються як атакам зловмисників, так і збоям інфраструктури, наприклад, відключенням живлення. Такі події можуть вплинути на роботу серверів доменних імен, зробити хмара недоступним або прямо порушити функціонування хмари.

Наприклад, атака на Akamai Technologies, проведена 15 червня 2004 р., викликала проблеми з дозволом доменних імен і великий збій, що торкнувся Google Inc., Yahoo! Inc. і багато інших сайтів. У травні 2009 р. Google виявився метою серйозної DoS-атаки (denial-of-service), що вивела з ладу на кілька днів такі сервіси як Google News і Gmail.

Блискавка викликала тривалий простий Amazon.com Inc. 29 і 30 червня 2012. Хмара Amazon Web Services (AWS) у східному регіоні Сполучених Штатів, що складає з десяти центрів даних у чотирьох зонах доступності, спочатку випробовувала проблеми через коливання електроживлення, імовірно, викликаних грозою. 29 червня 2012 гроза на східному узбережжі США вивела з ладу деяку апаратуру Amazon, розташовану у Віргінії, що порушило роботу компаній, що використовувала системи тільки із цього регіону. Як повідомляють, однієї з жертв цього простою виявився сервіс обміну фотографіями Instagram.

Відновлення після цих подій зажадало багато часу й було пов'язане з рядом проблем. Наприклад, один з десяти центрів не зміг перемкнутися на запасні генератори до того, як сіли джерела безперебійного живлення (UPS). AWS застосувало «площини керування» (control planes), щоб дозволити користувачам перемкнутися на ресурси в інших регіонах, але цей програмний компонент також відмовив.

Процес початкового завантаження виявився недосконалим і збільшив час, необхідне для перезапуску сервісів Elastic Compute Cloud (EC2) і Elastic Block Store (EBS). Ще однією критичною проблемою став «баг» в Elastic Load Balancing (ELB), що служив для маршрутизації трафіку на сервери з доступними ресурсами. Аналогічний «баг» порушив процес відновлення Relational Database Service (RDS). Ця подія виявила «сховані» проблеми, які можуть відбутися тільки при особливих обставинах.

Частини хмари

Провайдер хмарних додатків, провайдер хмарного сховища й провайдер мережі можуть реалізовувати різні політики. Непередбачена взаємодія між балансувальником навантаження й інших реактивних механізмів може привести до втрати динамічної стійкості. Непередбачене сполучення незалежних компонентів, керуючих навантаженням, споживанням енергії й елементами інфраструктури може привести до виникнення небажаного зворотного зв'язка й нестабільності, аналогічно тому, як це може відбутися при маршрутизації, заснованої на політиках, при використанні Internet Border Gateway Protocol (BGP).

Наприклад, балансувальник навантаження провайдеру додатків може взаємодіяти з оптимізатором споживання енергії провайдеру інфраструктури. Деякі з таких сполучень можуть виникати тільки в надзвичайних ситуаціях, і їх буває дуже складно виявити при роботі в нормальних умовах. Це може привести до катастрофічних наслідків, коли система буде намагатися відновитися після серйозного збою, як це й відбулося в 2012 р. з AWS.

Кластеризація ресурсів у центрах даних, розташованих у різних географічних областях, – один із засобів, застосовуваних для зниження ймовірності катастрофічних збоїв. Такий географічний розподіл ресурсів може мати додатковий позитивний побічний ефект. Воно дозволяє скоротити трафік обміну інформацією й витрати на електрику, переводячи обчислення в ті місця, де електроенергія дешевше. Також це може підвищити продуктивність за рахунок застосування інтелектуальної й ефективної стратегії балансування навантаження.

При організації хмарної інфраструктури ви повинні ретельно збалансувати мети системи, такі як максимальна пропускна здатність, використання ресурсів і фінансові переваги з потребами користувачів, такими як низькі витрати, низький час відгуку й максимальна доступність. За будь-яку оптимізацію системи доводиться платити ростом її складності. Наприклад, затримка при обміні інформацією через глобальну мережу (WAN) істотно більше, ніж у локальній мережі, і вимагає розробки нових алгоритмів глобального прийняття рішень.

Проблеми, що виникають при наданні хмарних послуг

Хмарні обчислення успадковують деякі із проблем від паралельних і розподілених обчислень. Але їм властиві й деякі власні проблеми. Конкретні проблеми відрізняються для трьох моделей надання хмарних послуг, але у всіх випадках вони обумовлені самою природою надання обчислювальних ресурсів як комунальних послуг, заснованого на спільному використанні й віртуалізації ресурсів, і потребуючу модель довіри, відмінну від повсюдно прийнятої моделі, орієнтованої на користувача, що довгий час була стандартом.

Сама значна проблема – безпека. Для майбутнього хмарної послуги надто важливо завоювати довіру великої кількості користувачів. Не можна розраховувати, що загальнодоступна хмара буде прийнятним середовищем для всіх додатків. Додатка підвищеної відповідальності, що управляють критично важливими інфраструктурами, додатка для охорони здоров'я й інші, швидше за все, будуть виконуватися в закритих хмарах.

Багато додатків, що працюють у реальному часі, також, швидше за все, помістять у закриті хмари. Деяким додаткам найкраще підійде гібридне хмарне середовище. Такі додатки можуть зберігати коштовні дані в закритій хмарі й використовувати загальнодоступну хмару для певних видів обробки.

У моделі Software as a Service (SaaS) виникають ті ж проблеми, що й в інших онлайнових послугах, що вимагає захисти особистої інформації, таких як фінансові або медичні послуги. У цьому випадку користувач взаємодіє із хмарними сервісами через чітко певний інтерфейс. Тому, у принципі, у

провайдеру сервісів виникає менше складностей з перекриттям деяких каналів атаки (attack channels).

Проте, такі сервіси уразливі для DoS-атак і зловмисних внутрішніх користувачів. Найбільш уразливі для атак дані в сховище, тому приділите особливу увагу захисту серверів зберігання. Реплікація даних, необхідна, щоб забезпечити безперервність обслуговування при відмові систем зберігання, збільшує уразливість. Шифрування даних може захистити дані при зберіганні, але, в остаточному підсумку, дані прийде розшифрувати для обробки. І тоді вони будуть уразливі для атаки.

Модель Infrastructure as a Service (IaaS), безсумнівно, сама складна з погляду захисту від атак. Справді, користувач IaaS має набагато більше волі, чим у двох інших моделях надання хмарних послуг. Додаткове джерело проблем – те, що чимала кількість хмарних ресурсів можна задіяти для атаки на мережу й інфраструктуру обчислень.

Украй важливою архітектурною особливістю цієї моделі є віртуалізація, але вона робить системи підданими новим видам атак. Довірена обчислювальна база (trusted computing base, TCB) віртуального середовища містить не тільки встаткування й гіпервізор, але й керуючу ОС. Можна зберегти стан всієї віртуальної машини (VM) у файл, щоб її було можна переносити й відновлювати – підтримка цих двох операцій дуже бажана.

Проте, ця можливість ускладнює стратегії по змісту серверів, що належать організації, у необхідному стабільному стані. Справді, заражена VM може бути неактивної під час перевірки систем. Потім вона почне працювати й заразить інші системи. Це – ще один приклад того, як у базових технологіях хмарних обчислень сполучаються корисні й шкідливі ефекти.

Наступна істотна проблема пов'язана з керуванням ресурсами хмари. Кожне стратегія систематичного керування ресурсами (на відміну від керування по ситуації) вимагає існування керуючих компонентів, призначених для реалізації декількох класів політик: керування доступом, виділення ресурсів,

балансування навантаження, оптимізації енергоспоживання й – останнє один по одному, але не по важливості – надання гарантій якості обслуговування (quality of service, QoS).

Щоб реалізувати ці політики, що управляють компоненти повинні мати точну інформацію про глобальний стан системи. Визначення стану складної системи з 106 або більше серверами, розподіленими по великій території, – нездійсненне завдання. Справді, зовнішнє навантаження, а також стан окремих ресурсів, дуже швидко міняються. У результаті керуючі компоненти повинні функціонувати в умовах неповного або приблизного знання про стан системи.

Здається розумним очікувати, що така складна система може функціонувати тільки на основі принципів самоврядування. Але самоврядування й самоорганізація підвищують вимоги до реалізації процедур ведення журналів і аудита, критично важливих для забезпечення безпеки й довіри до провайдеру хмарних обчислень.

При самоврядуванні стає майже неможливим ідентифікувати, з яких причин почате та або інша дія, через якого виник пролом у захисті.

Остання велика проблема, що я торкнуся, пов'язана із сумісністю й стандартизацією. Залежність від постачальника – той факт, що користувач «прив'язаний» до певного постачальника хмарних послуг – серйозна проблема для хмарних користувачів. Стандартизація забезпечує сумісність і, отже, у якімсь ступені, рятує від побоювань, що сервіс, критично важливий для великої організації, буде недоступним протягом тривалого часу.

Уводити стандарти в період, коли технологія ще розвивається, складно, і може виявитися контрпродуктивним, оскільки, можливо, це буде перешкоджати нововведенням. Важливо усвідомлювати складність проблем хмарних обчислень і розбиратися в цілому ряді технічних і соціальних проблем, що виникають при хмарних обчисленнях. Зусилля по переносу IT-операцій у загальнодоступні й закриті хмари, виправдаються в довгостроковій перспективі.

АНБ могло використовувати Logjam для атаки на VPN сервери

Безліч хмарних сервісів піддаються недавно знайденої в TLS-протоколі уразливості за назвою Logjam (CVE-2015-4000), повідомляє компанія по інформаційній хмарній безпеці Skyhigh.

Logjam дуже нагадує уразливість FREAK, однак відрізняється тим, що замість ініціювання зміни шифрів RSA на RSA_EXPORT в Logjam виробляється відкат протоколу Діффі-Хеллмана, використовуваного для одержання ключа для подальшого шифрування, до слабозахищеного рівня DHE_EXPORT. Уразливість може бути проексплуатована для здійснення атаки «людина по-середині», що дозволить одержати доступ до даних, що проходять через TLS-З'єднання.

Logjam вражає всі сервери, які підтримують 512-бітне експортне шифрування, а також всі сучасні браузери. Відповідно до експертів, більше 8% з Топ-мільйона web-сайтів, що використовують HTTPS, і більше 3% ресурсів, відображуваних у браузері як заслуговують довіри, піддані даної уразливості.

У ході атаки з експлуатацією Logjam зловмисники можуть знизити стійкість шифрування в мільйонів HTTPS, SSH і VPN серверів, які підтримують DHE_EXPORT і використовують для генерації ключа прості 512-розрядні групи початкових чисел Діффі-Хеллмана.

На думку фахівців Skyhigh, команда вчених може зламати 768-бітне просте число, а група хакерів, фінансована державою, може замахнутися на 1024-бітне. Злом єдиного, найпоширенішого 1024-бітного простого числа, використовуваного web-серверами, дозволить прослуховувати підключення до 18% з мільйона найбільш популярних HTTPS-сайтів. Експерти вважають, що АНБ могло використовувати Logjam для атаки на VPN сервери.

Злом хмарних сервісів: Соціальний інжиніринг у дії

Захисне програмне забезпечення сучасних комп'ютерів – антивіруси, файрволи, антишпигуни, антиспамові рішення, системи виявлення вторгнень і т.д. – виконує завдання будь-якої навіть найвищої складності, але всіма цими високотехнологічними рішеннями управляти набагато легше, ніж людьми. Як би

складним не було програмне забезпечення, будь-який фахівець із області інформаційної безпеки з легкістю підтвердить, що сама слабка ланка в будь-якій захисній системі – це людський фактор.

В останні роки в багатьох країнах одержав поширення соціальний інжиніринг, тобто метод керування діями людини без використання технічних засобів, заснована на використанні людських слабостей. Пояснення сутності соціального інжинірингу й, зокрема, таких його різновидів, як гіпноз і нейролінгвістичне програмування (НЛП), гідно окремої статті, саме головне, що ключем до розуміння уразливості людського фактора, є взаємодія між свідомістю й підсвідомістю. Люди вірять у те, що приймають рішення усвідомлено, але НЛП і гіпноз уже давно продемонстрували силу підсвідомості, а дослідження останнього років підтвердили, що підсвідоме прийняття рішень випереджає свідоме часом на 10 секунд.

На цьому засновані технології, що дозволяють маніпулювати людьми, щоб змусити їх виконати певні дії й тим самим розкрити конфіденційну інформацію. Розглянемо, як такі методи можуть бути використані для злому хмарних систем.

Приклади несанкціонованого доступу до даних

Дослідницькою групою були внесені зміни в роботу сайту – про це знало керівництво компанії, який належав сайт, але IT-фахівці не були попереджені. Для того щоб користувач міг працювати з електронною поштою, на додаток до звичайної вимоги відповістити на секретне запитання система попросила ввести ім'я користувача й пароль. Як результат, протягом години було уведено 25% логінів/паролів. Крім того, пройшло 2 години, перш ніж IT-команда зміркувала, що відбувається, і ще 2 години, перш ніж фахівці компанії змогли опублікувати попередження для користувачів. Незважаючи на вжиті заходи, облікових даних користувачів надходили ще кілька годин після того, як було розіслане попередження.

В іншому випадку дослідницька група зареєструвала домен, дуже схожий на домен цільової компанії. Потім всім співробітникам компанії був розісланий

лист, що описує нову процедуру уведення пароля, у результаті чого співробітники попадали на фальшивий сайт компанії. Адреса підробленого сайту, що містить схожий з реальним домен, відкрито розміщався в листі; дослідники не робили спроб замаскувати URL. Як і передбачалося, більшість співробітників, що одержали лист, слухняно виконали зазначені дії.

Ще одне дослідження показало, що наявність строгих процедур у процесі внесення змін в адміністративний інтерфейс також є уразливим місцем системи безпеки через відсутність гнучкості. Тобто, якщо хакеру вдається реалізувати позаштатну ситуацію, користувач, швидше за все, буде шукати шляхи до її виправлення, і зловмисникові залишиться тільки «підказати» йому відповідь. Таким чином, часто строгі алгоритми роботи системи відкривають особам, які у достатній мері володіють методами соціального інжинірингу, доступ до конфіденційної інформації компанії.

У цілому експерименти показали, що користувачі не просто не бачать різниці між офіційним і підробленим доменом, проблема набагато глибше. Незважаючи на регулярно проведені в компаніях тренінги й впроваджувані навчальні програми, більшість співробітників воліє знайомитися не з усіма методами виявлення й протидії вторгненням, а лише з тими, які для них важливі, приміром, особливо популярні можливості безпечного онлайн-банкинга й роботи з відомими онлайн-аукціонами. Таким чином, співробітники компаній своїми діями підтвердили, що не очікують фішинг-атак, спрямованих на їхні робітники дані, хоча й знають, що така атака можлива.

Лов на живця

Зрозуміло, розроблювачі програмного забезпечення не можуть повною мірою враховувати людський фактор, але для того, щоб визначити найбільш уразливі місця в системі безпеки окремо взятої компанії, як ні парадоксально, не обійтися знову ж без допомоги користувачів. Мова йде про імітацію хакерських атак на систему замовника, результати яких дадуть клієнтові механізм для виміру ефективності поточного захисту системи від соціального інжинірингу. Під

атакою, таким чином, розуміються різні тестові сценарії, які призначені для оцінки успішності/неуспішності системи.

Певні труднощі представляє розробка подібних атак, оскільки, по-перше, атакувати потрібно не якого-небудь із користувачів, а безпосередньо службу підтримки, змушуючи її надати хакеру права адміністратора, а по-друге, потрібно досить точно й швидко визначити сектор системи для перевірки, щоб не затягувати процес і швидше ліквідувати уразливість.

Злом хмарного сервісу

Здійснювану в дослідницьких цілях атаку можна розділити на кілька фаз. Протягом першої – відбувається збір початкових даних про систему, приміром, про розташовані в інших країнах підрозділах, про територіальне покриття службою підтримки, інформації про клієнтів, а також збір відомостей про топ-менеджерів і директорів компанії.

Друга фаза містить у собі доступ до мереж компанії в нічний час доби, для того щоб оцінити, що відбувається із системою безпеки у відсутності адміністраторів, а також для того, щоб при необхідності змінити IP-адреса. Також у другій фазі відбувається збір даних про недавно зроблені звільнення й скорочення співробітників, а заодно перевіряється, чи внесли адміністратори відповідні зміни в систему. На третьому етапі дослідниками беруться під контроль облікові записи керуючих і технічного директорів цільової компанії. На четвертому здійснюється контакт зі співробітниками, щоб домогтися їхнього особистого розташування.

Результат експериментальної атаки перевершив всі очікування дослідників: вони домоглися бажаного всього за три цілеспрямованих телефонних дзвінка з використанням психологічних методів соціального інжинірингу в підрозділи компанії – у Великобританії, США й у Китаї. Спочатку керуючий директор подзвонив у британський офіс компанії незадовго до закінчення робочого дня, попередив про можливі проблеми й запросив інформацію про доступ до підтримки, але, відпрацьовуючи сценарій

зловмисника, не просив відкрити йому доступ у систему. Другий дзвінок надійшов в офіс у США від клієнта керуючого директора, що просив внести зміни в логін і пароль, пояснюючи це тим, що в нього виникли труднощі з доступом у систему. Третій дзвінок надійшов у китайський офіс компанії від технічного директора компанії, що перебуває в надзвичайних обставинах.

Внаслідок атаки дослідникам удалося обійти строгі процедури забезпечення безпеки; співробітники компанії представили дослідникам подробиці облікових записів; трафік удалося перенаправляти на нову адресу, і в цьому був задіяний співробітник компанії; по телефоні був продиктований новий пароль адміністратора. Надзвичайно важливо, що ніхто зі співробітників не спробував установити особистість що дзвонила.

Учасники експерименту, з огляду на характер атаки, відсутність аудита й повідомлень про зміни в системі, прийшли до виводу, що в результаті вторгнення реальних хакерів несанкціонований доступ міг залишатися непоміченим протягом тривалого часу. Таким чином, незважаючи на те, що хмарна система є технічним рішенням, подібні сервіси залежать від людського фактора так само, як більшість систем інформаційної безпеки. Як підтвердив експеримент, соціальний інжиніринг виявився найшвидшим і легенею шляхом до порушення інформаційної безпеки й самим трудновизначаємим. Методи соціально інжинірингу, на відміну від електронного злому сайтів, майже неможливо каталогізувати, а вплив доводиться оцінювати, покладаючись на показання свідків співробітників, що стали жертвами, причому треба враховувати, що вони можуть псувати реальність, бажаючи врятувати свою репутацію.

3.2 Розробка структурної схеми

У список найпоширеніших застосувань хмарних сервісів входять: зберігання сканів паспорта й інших особистих документів; синхронізація бази паролів, контактів, листів; створення сайтів; зберігання версій вихідних кодів і

т.д. Коли хмарний сервіс зберігання даних Dropbox повідомив про закриття уразливості в генераторі посилань, в інтернеті знову заговорили про те, як важливо шифрувати конфіденційні дані, перш ніж викладати їх на який-небудь ресурс, навіть якщо він приватний. Шифрування файлів (FLE) дійсно дозволить забезпечити захист конфіденційної інформації в хмарі, навіть у випадку виявлення уразливостей контролю доступу до документів користувачів у тім або іншому хмарному сервісі.

Може зложитися враження, що якщо не викладати в хмари секретні дані, або їх шифрувати, то й ризиків ніяких не буде. Чи не так це? Як виявилося, не зовсім.

В інтернеті часто зустрічаються рекомендацій з «ефективного використання хмарних файлових сервісів», наприклад – інструкції з вилученого керування комп'ютером, стеженню за комп'ютером під час своєї відсутності, керування torrent-завантаженнями й багато інше. Інакше кажучи, користувачі самі створюють усілякі діри, якими з легкістю скористається й троянець, і хробак, і тим більше хакер, особливо якщо мова йде про цільові атаки.

Ми задалися питанням – наскільки великий ризик зараження корпоративної мережі через хмарний сервіс?

Спочатку, використовуючи фішинг, розроблювач заразив ноутбук співробітника, далі – впровадив шкідливі скрипти в документи, що зберігаються в «хмарній» папці ноутбука. Dropbox автоматично обновив (синхронізував) заражені документи на всіх пристроях, пов'язаних з аккаунтом користувача. Щодо цього Dropbox не унікальний – функція автоматичної синхронізації є у всіх популярних додатках для доступу до хмарних файлових сервісів, у тому числі Onedrive (він же Skydrive), Google Disk, Yandex Disk і т.д.

Коли користувач відкрив заражений документ на робочому комп'ютері, що перебуває в корпоративній мережі, впроваджені в документ скрипти встановили в систему бекдор DropSmack, створений розроблювачем спеціально для цього пен-тесту. Як можна догадатися за назвою, ключова особливість

DropSmack – використання хмарного файлового сховища Dropbox як канал для керування бекдором і передачі корпоративних документів назовні крізь корпоративний файервол.

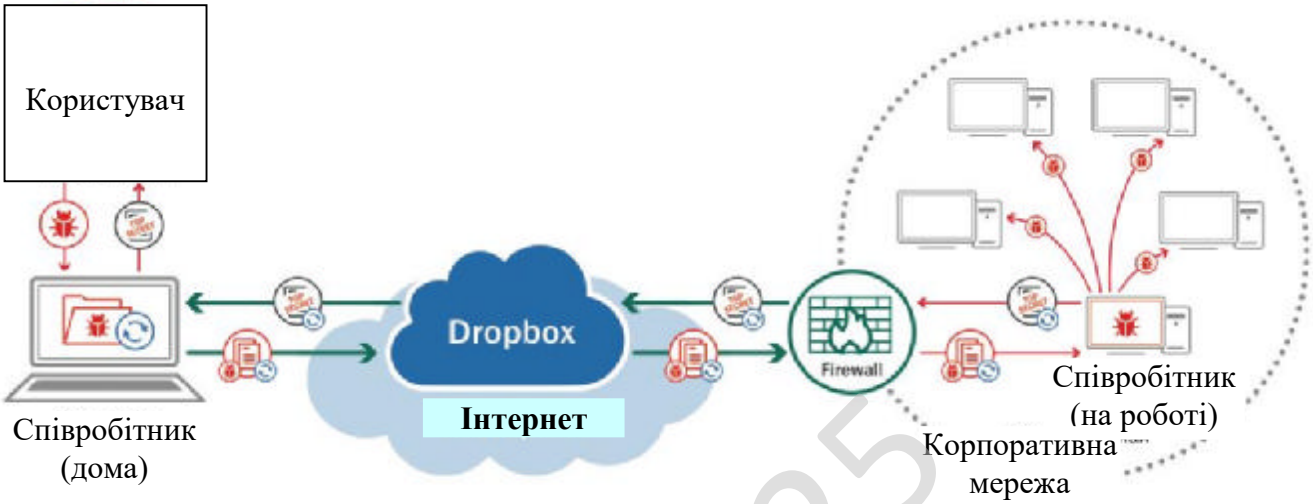

Рисунок 3.1 – Структурна схема системи

Метод проникнення в корпоративну мережу, використаний розроблювачем у ході пен-тесту, дивує своєю простотою – так, це очевидна діра!

Ми вирішили перевірити, чи використовують справжні зловмисники Dropbox, OneDrive, Yandex Disk і Google Disk для поширення шкідливого ПЗ. Зібравши інформацію з KSN про детектах вірусів, виявлених в «хмарних» папках на комп'ютерах користувачів, ми з'ясували, що такі зараження зафіксовані в дуже невеликого числа користувачів: у травні цього року тільки 8,7 тис. чоловік зштовхнулися із зараженням «хмарної» папки. У користувачів домашніх продуктів ПК на такі шкідливі програми доводиться 0,42% всіх детектів, у користувачів корпоративних продуктів – 0,24%.

Слід зазначити одну немаловажну деталь – якщо вірус потрапив у хмару з одного пристрою, всі підключені до зараженого аккаунту клієнти самі завантажать його на свої пристрої, так ще й по протоколі HTTPS. Навіть якщо на одному із пристроїв антивірус помітить і видалить заразу прямо в папці

синхронізації, хмарний клієнт буде, з обов'язку служби, боротися зі сформованої рассинхронізацією, без кінця завантажуючи вірус із хмари.

За нашим даними, близько 30% шкідливого ПЗ, виявленого в «хмарних» папках на домашніх комп'ютерах, попадає на комп'ютери через механізми синхронізації! У корпоративних користувачів цей показник досягає 50%. Таким чином, механізм, що використовувався демонстраційним вірусом розроблювача, приводить до заражень у реальному житті. На щастя, ми поки не виявили цільових атак з використанням хмарних сервісів зберігання даних.

Серед шкідливого ПЗ, виявленого нами в «хмарних» папках на комп'ютерах користувачів, переважають файли форматів Win32, MSIL, VBS, PHP, JS, Excel, Word, Java. Варто відзначити, що між корпоративним і домашнім користувачами є невелика різниця – у перших частіше зустрічаються заражені файли MS Office, у других у списку є унікальні звірі – шкідливі Android-додатка.

Найчастіше вирусописувачі використовують хмарні сховища не як платформу для поширення, а в якості хостингу для шкідливих програм – у ході дослідження ми не зустріли жодного хробака або бекдора (не вважаючи DropSmack), спеціально націленого на хмарні файлові сховища. Звичайно, самі сервіси намагаються активно боротися зі шкідливими програмами, які займають вільне місце в хмарі. Крім того, хостинг шкідливих програм негативно впливає на репутацію сервісу, хоча формально хмарні сервіси й не несуть відповідальності за те, які файли завантажуються клієнтами в сховище. Очевидно, що регулярне сканування всіх файлів, що втримуються в хмарі, зажадає занадто багато ресурсів, які сервісам вигідніше використовувати для зберігання даних.

Підсумком проведеного дослідження стало розуміння, що ризик зараження корпоративної мережі через хмарне сховище порівняно невеликий – протягом року заразитися ризикує 1 з 1000 корпоративних користувачів, що використовують хмарні сервіси. Однак треба враховувати, що в деяких випадках навіть одиничне зараження комп'ютера в корпоративній мережі може привести до значного збитку.

Можливі міри захисту корпоративної мережі:

– Підсилити захист в Firewall/IDS, заблокувати доступ до серверів відомих сервісів. Більшим мінусом такого підходячи є більша ресурсоємність – необхідно уважно стежити, чи не з'явилися нові кандидати в «чорний список».

– Установити багатофункціональний Security Suite, що включає в себе евристичний і поведінковий антивірус, функції обмеження доступу (HIPS), контроль за роботою операційної системи (System Watcher або Hypervisor), захист від експлуатації уразливостей і т.д. При цьому необхідно ретельно все настроїти.

– Тому що навіть самий навернений Security Suite може пропустити APT, варто звернути увагу на технологію Контроль додатків (у режимі «Заборонено за замовчуванням»). Це, мабуть, один із самих надійних засобів, що дозволяє заблокувати будь-яке невідоме ПЗ (у тому числі розповсюджуване в ході цільової атаки). Саме складне завдання, що виникає під час впровадження Контролю додатків, – настроїти відповідні правила, щоб всі дозволені додатки запускалися й обновлялися без проблем. Для цього виробниками продуктів з функцією Контролю додатків були розроблені спеціальні інструменти – функція відновлення ПЗ через довірені програми відновлення, передвстановлені списки білого ПЗ, що включають всі можливі системні й користувальницькі файли, доступ до величезних хмарних сервісів і баз інформації про все різноманіття білого ПЗ.

– В особливих випадках, варто використовувати «Контроль додатків», щоб обмежити використання хмарних клієнтів у корпоративній мережі, тобто дозволити запуск додатків синхронізації «хмарної» папки тільки довіреним співробітникам.

Ну, а для самих-самих закритих систем, таких, які управляють роботою електростанцій, систем водопостачання, зберігають держтайну або дозволяють запускати міжконтинентальні ракети – настійно рекомендуємо повністю відмовити від використання хмарних файлових сховищ.

3.3 Розробка функціональної схеми

Для реалізації системи виявлення уразливих додатків у Cloud-сервісах використаємо нейронну мережу. Наведемо опис цієї нейромережі.

Нейрокомп’ютерна мережа зустрічного поширення

Дослідимо **нейрокомп’ютерну мережу зустрічного поширення** У процесі навчання вхідні вектори асоціюються з відповідними вихідними векторами. Ці вектори можуть бути двійковими, що складаються з нулів і одиниць, або безперервними. Коли мережа навчена, прикладання вхідного вектора приводить до необхідного вихідного вектора. Узагальнююча здатність мережі дозволяє одержувати правильний вихід навіть при додатку вхідного вектора, що є неповним або злегка невірним. Це дозволяє використовувати дану мережу для розпізнавання образів, відновлення образів і посилення сигналів.

Структура мережі

На рисунку 3.2 показана спрощена версія прямої дії мережі зустрічного поширення. На ньому ілюструються функціональні властивості цієї парадигми.

Нейрони шару 0 (показані кружками) служать лише крапками розгалуження й не виконують обчислень. Кожний нейрон шару 0 з'єднаний з кожним нейроном шару 1 (називаного шаром Кохонена) окремою вагою $w_{mn.}$ Ці ваги в цілому розглядаються як матриця ваг $\boldsymbol{W}$. Аналогічно, кожний нейрон у шарі Кохонена (шар 1) з'єднаний з кожним нейроном у шарі Гроссберга (шар 2) вагою $v_{пр}$. Ці ваги утворять матрицю ваг **V**. Все це досить нагадує інші мережі, розходження, однак, складається в операціях, виконуваних нейронами Кохонена й Гроссберга.

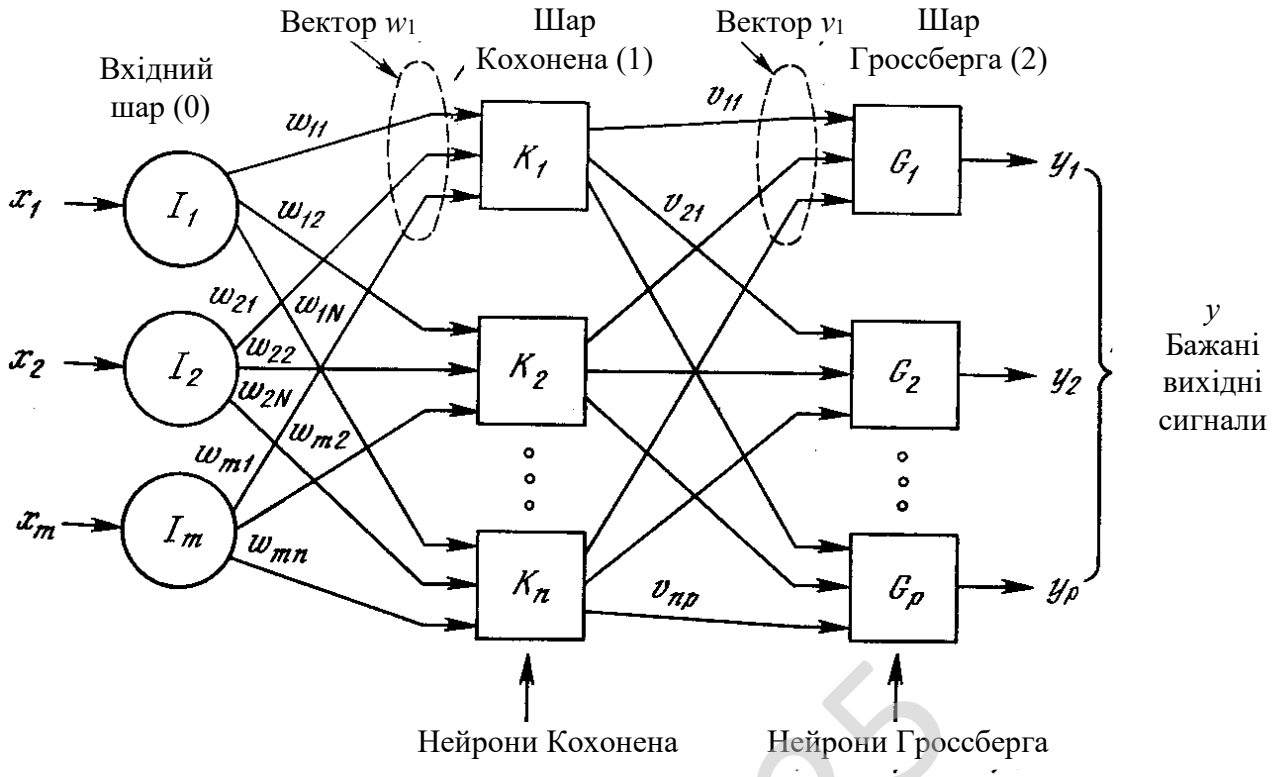

Рисунок 3.2 – Мережа із зустрічним розпізнаванням без зворотних зв'язків

Як і багато інших мереж, зустрічне поширення функціонує у двох режимах: у нормальному режимі, при якому приймається вхідний вектор **X** и видається вихідний вектор **Y**, і в режимі навчання, при якому подається вхідний вектор і ваги коректуються, щоб дати необхідний вихідний вектор.

Нормальне функціонування

Шари Кохонена

У своїй найпростішій формі шар Кохонена функціонує в дусі «переможець забирає все», тобто для даного вхідного вектора один і тільки один нейрон Кохонена видає на виході логічну одиницю, всі інші видають нуль. Нейрони Кохонена можна сприймати як набір електричних лампочок, так що для будь-якого вхідного вектора загоряється одна з них.

Асоційоване з кожним нейроном Кохонена множина ваг з'єднує його з кожним входом. Наприклад, на рисунку 3.9 нейрон Кохонена K_1 має ваги w_{11}, w_{21}, …, w_{m1}, які складають вектор ваги $\mathbf{W}_1$. Вони з'єднуються через вхідний шар із вхідними сигналами x_1, x_2, …, x_m, , які складають вхідний вектор X. Подібно

нейронам більшості мереж вихід NET кожного нейрона Кохонена є просто сумою зважених входів. Це може бути виражене в такий спосіб:

$$NET_j = w_{1j}x_1 + w_{2j}x_2 + \ldots + w_{mj}x_m \quad (3.1)$$

де NET_j – це вихід NET нейрона Кохонена j,

$$NET_j = \sum_i x_i w_{ij} \quad (3.2)$$

або у векторному запису

$$N = XW, \quad (3.3)$$

де N – вектор виходів NET шару Кохонена.

Нейрон Кохонена з максимальним значенням NET є «переможцем». Його вихід дорівнює одиниці, в інших він дорівнює нулю.

Шар Гроссберга

Шар Гроссберга функціонує в подібній манері. Його вихід NET є зваженою сумою виходів $k_1, k_2, \ldots, k_n$ шару Кохонена, що утворюють вектор **К.** Вектор з'єднуючих ваг, позначений через **V**, складається з ваг $v_{11}, v_{21}, \ldots, v_{np}$. Тоді вихід NET кожного нейрона Гроссберга є

$$NET_j = \sum_i k_i w_{ij}, \quad (3.4)$$

де NET_j – вихід j-го нейрона Гроссберга, або у векторній формі

$$Y = KV, \quad (3.5)$$

де Y – вихідний вектор шару Гроссберга,

К – вихідний вектор шару Кохонена,

V – матриця ваг шару Гроссберга.

Якщо шар Кохонена функціонує таким чином, що лише в одного нейрона величина NET дорівнює одиниці, а в інших дорівнює нулю, то лише один елемент вектора К відмінний від нуля, і обчислення дуже прості. Фактично кожний нейрон шару Гроссберга лише видає величину ваги, що зв'язує цей нейрон з єдиним ненульовим нейроном Кохонена.

Навчання шару Кохонена

Шар Кохонена класифікує вхідні вектори в групи схожих. Це досягається за допомогою такого підстроювання ваг шару Кохонена, що близькі вхідні

вектори активують той самий нейрон даного шару. Потім задачею шару Гроссберга є одержання необхідних виходів.

Навчання Кохонена є самонавчанням, що протікає без учителя. Тому важко (і не потрібно) пророкувати, який саме нейрон Кохонена буде активуватися для заданого вхідного вектора. Необхідно лише гарантувати, щоб у результаті навчання розділялися несхожі вхідні вектори.

Попередня обробка вхідних векторів

Досить бажано (хоча й не обов'язково) нормалізувати вхідні вектори перед тим, як пред'являти їхньої мережі. Це виконується за допомогою ділення кожного компонента вхідного вектора на довжину вектора. Ця довжина перебуває добуванням квадратного кореня із суми квадратів компонент вектора. В алгебраїчному записі

$$x'_i = \frac{x_i}{\sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}} \qquad (3.6)$$

Це перетворює вхідний вектор в одиничний вектор з тим же самим напрямком, тобто у вектор одиничної довжини в n-мірному просторі.

Рівняння (3.6) узагальнює добре відомий випадок двох вимірів, коли довжина вектора дорівнює гіпотенузі прямокутного трикутника, утвореного його *x* и *в* компонентами, як це треба з відомої теореми Піфагора. На рисунку 3.3 такий двовимірний вектор V представлений у координатах *x-y,* причому координата *x* дорівнює чотирьом, а координата *y* – трьом. Квадратний корінь із суми квадратів цих компонентів дорівнює п'яти. Ділення кожного компонента V на п'ять дає вектор V з компонентами 4/5 і 3/5, де V' указує в тому же напрямку, що й V, але має одиничну довжину.

На рисунку 3.4 показано кілька одиничних векторів. Вони кінчаються в крапках одиничної окружності (окружності одиничного радіуса), що має місце, коли в мережі лише два входи. У випадку трьох входів вектори представлялися б стрілками, що кінчаються на поверхні одиничної сфери. Ці подання можуть бути перенесені на мережі, що мають довільне число входів, де кожний вхідний вектор

є стрілкою, що кінчається на поверхні одиничної гіперсфери (корисною абстракцією, хоча й не безпосередньої візуалізації, що допускає).

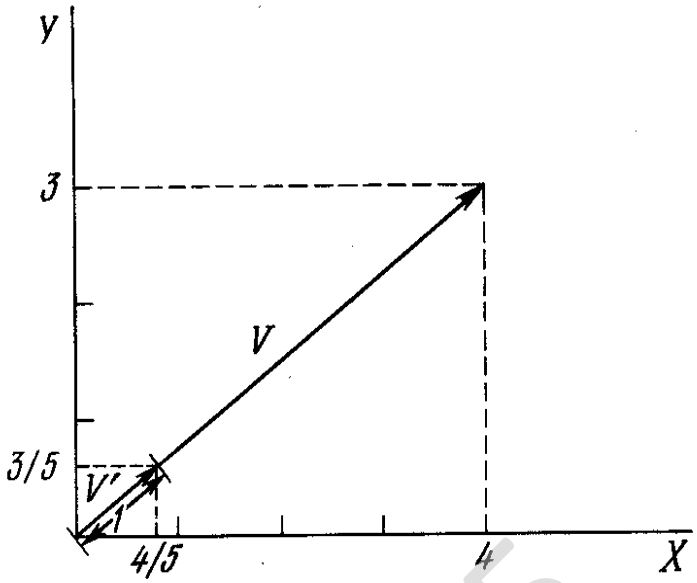

Рисунок 3.3 – Одиничний вхідний вектор

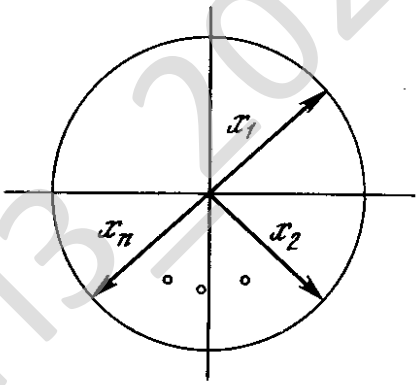

Рисунок 3.4 – Двовимірні одиничні вектори на одиничній окружності

При навчанні шару Кохонена на вхід подається вхідний вектор і обчислюються його скалярні добутки з векторами ваг, зв'язаними з усіма нейронами Кохонена. Нейрон з максимальним значенням скалярного добутку оголошується «переможцем» і його ваги підбудовуються. Так як скалярний добуток, використовуваний для обчислення величин NET, є мірою подібності між вхідним вектором і вектором ваг, то процес навчання складається у виборі нейрона Кохонена з ваговим вектором, найбільш близьким до вхідного вектора, і подальшому наближенні вагового вектора до вхідного. Знову відзначимо, що

процес є самонавчанням, виконуваним без учителя. Мережа самоорганізується таким чином, що даний нейрон Кохонена має максимальний вихід для даного вхідного вектора. Рівняння, що описує процес навчання має такий вигляд:

$$w_н = w_c + \alpha(x - w_c), \quad (3.7)$$

де $w_н$ – нове значення ваги, що з'єднує вхідний компонент x з нейроном, що виграв;

w_c – попереднє значення цієї ваги;

α – коефіцієнт швидкості навчання, що може варіюватися в процесі навчання.

Кожна вага, пов'язана з нейроном, що виграв, Кохонена, змінюється пропорційно різниці між його величиною й величиною входу, до якого він приєднаний. Напрямок зміни мінімізує різниця між вагою і його входом.

На рисунку 3.5 цей процес показаний геометрично у двовимірному виді. Спочатку знаходиться вектор $X - W_c$, для цього проводиться відрізок з кінця W у кінець X. Потім цей вектор коротшає множенням його на скалярну величину α, меншу одиниці, у результаті чого виходить вектор зміни δ. Остаточно новий ваговий вектор $W_н$ є відрізком, спрямованим з початку координат у кінець вектора δ. Звідси можна бачити, що ефект навчання складається в обертанні вагового вектора в напрямку вхідного вектора без істотної зміни його довжини.

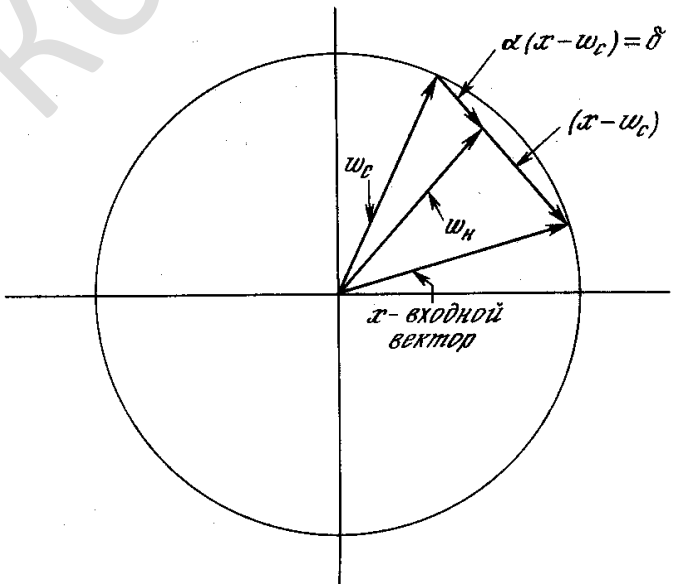

Рисунок 3.5 – Обертання вагового вектора в процесі навчання

$W_н$ – вектор нових вагових коефіцієнтів,

$W_с$ – вектор старих вагових коефіцієнтів

Змінна *к* є коефіцієнтом швидкості навчання, що спочатку звичайно дорівнює ~ 0,7 і може поступово зменшуватися в процесі навчання. Це дозволяє робити більші початкові кроки для швидкого грубого навчання й менші кроки при підході до остаточної величини.

Якби з кожним нейроном Кохонена асоціювався один вхідний вектор, то шар Кохонена міг би бути навчений за допомогою одного обчислення на вагу. Ваги нейрона-переможця прирівнювалися б до компонентів навчального вектора ($\alpha = 1$). Як правило, множина, що навчає, включає багато подібних між собою вхідних векторів, і мережа повинна бути навчена активувати один й той самий нейрон Кохонена для кожного з них. У цьому випадку ваги цього нейрона повинні виходити усередненням вхідних векторів, які повинні його активувати. Поступове зменшення величини (зменшує вплив кожного навчального кроку, так що остаточне значення буде середньою величиною від вхідних векторів, на яких відбувається навчання. Таким чином, ваги, асоційовані з нейроном, приймуть значення поблизу «центра» вхідних векторів, для яких даний нейрон є «переможцем».

Вибір початкових значень вагових векторів

Всім вагам мережі перед початком навчання варто надати початкові значення. Загальноприйнятою практикою при роботі з нейронними мережами є присвоювання вагам невеликих випадкових значень. При навчанні шару Кохонена випадково обрані вагарні вектори варто нормалізувати. Остаточні значення вагових векторів після навчання збігаються з нормалізованими вхідними векторами. Тому нормалізація перед початком навчання наближає вагові вектори до їхніх остаточних значень, скорочуючи, таким чином, що навчає процес.

Рандомизація ваг шару Кохонена може породити серйозні проблеми при навчанні, так як в результаті її вагові вектори розподіляються рівномірно по

поверхні гіперсфери. Через те, що вхідні вектори, як правило, розподілені нерівномірно й мають тенденцію групуватися на відносно малій частині поверхні гіперсфери, більшість вагових векторів будуть так вилучені від будь-якого вхідного вектора, що вони ніколи не будуть давати найкращої відповідності. Ці нейрони Кохонена будуть завжди мати нульовий вихід і виявляться марними. Більше того, ваг, що залишилися, які дають найкращі відповідності, може виявитися занадто мало, щоб розділити вхідні вектори на класи, які розташований близько друг до друга на поверхні гіперсфери.

Допустимо, що є кілька множин вхідних векторів, всі множини подібні, але повинні бути розділені на різні класи. Мережа повинна бути навчена активувати окремий нейрон Кохонена для кожного класу. Якщо початкова щільність вагових векторів в околиці навчальних векторів занадто мала, то може виявитися неможливим розділити подібні класи через те, що не буде достатньої кількості вагових векторів в околиці, яка має для нас інтерес, щоб приписати по одному з них кожному класу вхідних векторів.

Навпаки, якщо кілька вхідних векторів отримані незначними змінами з того самого зразка й повинні бути об'єднані в один клас, то вони повинні включати той самий нейрон Кохонена. Якщо ж щільність вагових векторів дуже висока поблизу групи злегка різних вхідних векторів, то кожний вхідний вектор може активувати окремий нейрон Кохонена. Це не є катастрофою, так як шар Гроссберга може відобразити різні нейрони Кохонена в той самий вихід, але це марнотратна витрата нейронів Кохонена.

Найбільш бажане рішення полягає в тому, щоб розподіляти вагові вектори відповідно до щільності вхідних векторів, які повинні бути розділені, поміщаючи тим самим більше вагових векторів в околиці великої кількості вхідних векторів. На практиці це нездійсненно, однак існує кілька методів наближеного досягнення тих же цілей.

Одне з рішень, відоме за назвою *методу опуклої комбінації* (convex combination method), полягає в тому, що всі ваги прирівнюються до однієї й тієї же величині

$$w_i = \frac{1}{\sqrt{n}}, \tag{3.8}$$

де n – число входів і, отже, число компонентів кожного вагового вектора.

Завдяки цьому всі вагові вектори збігаються й мають одиничну довжину. Кожному ж компоненту входу **X** надається значення

$$x_i = \alpha x_i + \frac{1-\alpha}{\sqrt{n}}, \tag{3.9}$$

де n – число входів.

На початку α дуже мало, внаслідок чого всі вхідні вектори мають довжину, близьку до $1/\sqrt{n}$, і майже збігаються з векторами ваг. У процесі навчання мережі α поступово зростає, наближаючись до одиниці. Це дозволяє розділяти вхідні вектори й остаточно приписує їм їхні справжні значення. Вагові вектори відслідковують одну або невелику група вхідних векторів і наприкінці навчання дають необхідну картину виходів. Метод опуклої комбінації добре працює, але сповільнює процес навчання, так як вагові вектори підбудовуються до мети, що змінюється. Інший підхід складається в додаванні шуму до вхідних векторів. Тим самим вони піддаються випадковим змінам, схоплюючи зрештою ваговий вектор. Цей метод також працездатний, але ще більше медленен, чим метод опуклої комбінації.

Третій метод починає з випадкових ваг, але на початковій стадії навчального процесу підбудовує всі ваги, а не тільки пов'язані з нейроном Кохонена, що виграв. Тим самим вагові вектори переміщаються ближче до області вхідних векторів. У процесі навчання корекція ваг починає вироблятися лише для найближчих до переможця нейронів Кохонена. Цей радіус корекції поступово зменшується, так що зрештою коректуються тільки ваги, пов'язані з нейроном Кохонена, що виграв.

Ще один метод наділяє кожний нейрон Кохонена «Почуттям справедливості». Якщо він стає переможцем частіше своєї законної частки часу (приблизно $1/k$, де k – число нейронів Кохонена), він тимчасово збільшує свій поріг, що зменшує його шанси на виграш, даючи тим самим можливість навчатися й іншим нейронам.

У багатьох додатках точність результату істотно залежить від розподілу ваг. На жаль, ефективність різних рішень вичерпним образом не оцінена й залишається проблемою.

Розробка нейрокомп'ютерной мережі в застосуванні до виявлення уразливих додатків у Cloud-сервісах

При рішенні задачі виявлення уразливих додатків у Cloud-сервісах ми вибрали варіант у якому на етапі навчання в нас є наступні вхідні дані: відкритий текст, ключ, алгоритм шифрування, закритий текст.

На етапі виявлення уразливих додатків у Cloud-сервісах в нас є тільки закритий текст.

На етапі розпізнавання відбувається виділення із криптотекста знайомих системі зразків і подання їхнім одним нейроном або нейронним ансамблем на наступних рівнях. Як при навчанні, так і при розпізнаванні вхідні вектора є нечіткими, тобто є невеликий розкид векторів, що належать до одного класу. У зв'язку із цим нейромережа, що здійснює цю операцію, повинна мати певну здатність до статистичного усереднення. Навпроти, може виявитися, що група векторів перебуває в безпосередній близькості друг до друга, але всі вони представляють різні класи. Тоді нейромережа повинна визначати тонкі розходження між векторами. Ще одна вимога до нейромережі низького рівня обробки сигналу – навчання без учителя, тобто здатність самостійно розділяти вхідні сигнали на класи.

Велика кількість нейромережних алгоритмів виконують функцію поділу вхідного криптотекста на класи.

Відомі 3 математичні моделі цього поділу:

1. Поділ вхідних даних гіперплощинами (простий персептрон).

Застосування цього алгоритму виправдано тільки для задач, що володіють високою лінійністю. Наприклад, можна побудувати нейромережу, що розбиває крапки (0,0) і (1,1) на два класи для двовимірного сигналу, але неможливо вирішити задачу по розбивці крапок (0,0), (1,1) – перший клас, і (0,1), (1,0) – другий. Це широко відомий приклад нездатності простого персептрона вирішити задачу «або що виключає»

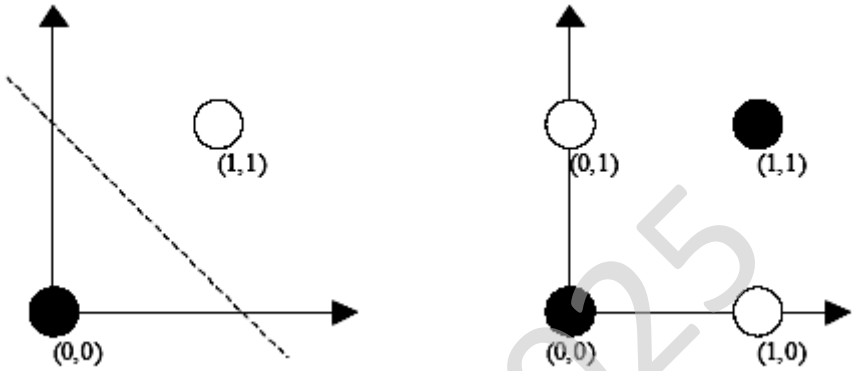

Рисунок 3.6– Теорема Мінського

2. Поділ вхідних даних гіперповерхнями (багатошарові персептрони).

При послідовному з'єднанні шарів, подібних простому персептрону, з'являється можливість комбінувати гіперплощини й одержувати гіперповерхні досить складної форми, у тому числі й замкнуті. Така нейромережа у принципі при достатнім числі нейронів здатна розділяти сигнали на класи практично будь-якої складності. Але застосування таких нейромереж обмежене складністю їхнього навчання. Був розроблений потужний алгоритм, називаний «алгоритмом зворотного поширення помилки», але й він вимагає значного часу навчання й не гарантує мінімального значення помилки (небезпека влучення в локальні мінімуми).

3. Пошук найбільшої відповідності (найменшого кутового або лінійного стану). При нормалізованих векторах входу, усі вектора розташовуються на поверхні гіперсфери.

Існує модель нейромережі, що відповідає цим вимогам – це мережа зустрічного поширення. В оригіналі вона являє собою об'єднання двох гарно відомих алгоритмів: карти Кохонена, що самоорганізується, й шару Гроссберга. У процесі навчання вхідні вектори асоціюються з відповідними вихідними векторами. Коли мережа навчена, додаток вхідного вектора приводить до необхідного вихідного вектора. Узагальнююча здатність мережі дозволяє одержувати правильний вихід навіть при додатку вхідного вектора, що є неповним або злегка невірним.

Схематично мережа зустрічного напрямку зображена на рисунку 3.7

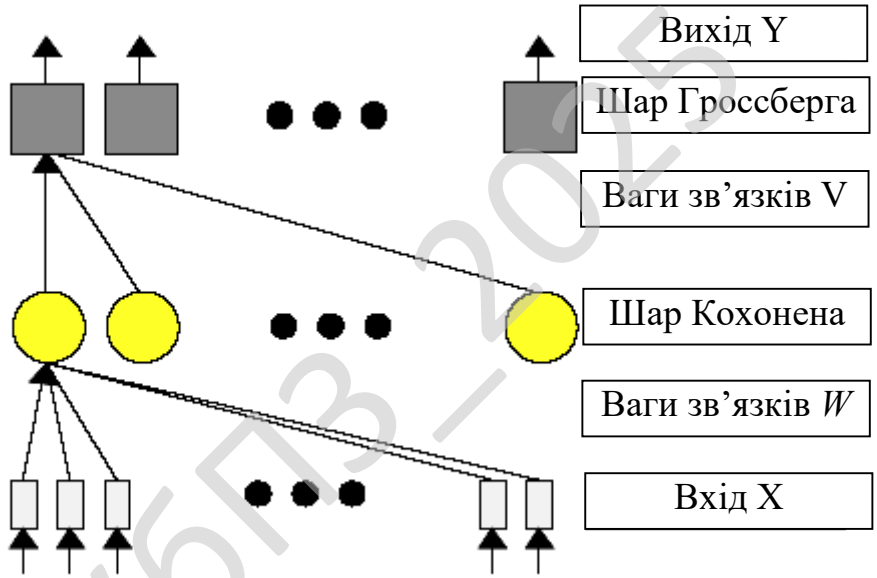

Рисунок 3.7– Мережа зустрічного поширення

Поширення даних у такій мережі відбувається в такий спосіб: вхідний вектор нормується на 1.0 і подається на вхід, що розподіляє його далі через матрицю ваг *W*. Кожний нейрон у шарі Кохонена обчислює суму на своєму вході й залежно від стану навколишніх нейронів цього шару стають активні або неактивним (рівні 1.0 і 0.0). Нейрони цього шару функціонують за принципом конкуренції, тобто в результаті певної кількості ітерацій активним залишається один нейрон або невелика група, «пухирець активності». Цей механізм

називається латеральним гальмуванням і докладно розглянутий у багатьох джерелах. Так як відпрацьовування цього механізму вимагає значних обчислювальних ресурсів, у даній моделі він замінений знаходженням нейрона з максимальною активністю й присвоєнням йому активності 1.0, а всім іншим нейронам 0.0. Таким чином, спрацьовує нейрон, для якого вектор входу ближче всього до вектора ваг зв'язків.

Якщо мережа перебуває в режимі навчання, то для нейрона, що виграв, відбувається корекція ваг матриці зв'язку за формулою

$$w_н = w_c + \alpha(x - w_c), \quad (3.9)$$

де $w_н$ – нове значення ваги,

w_c – старе значення,

α – швидкість навчання,

x – величина входу.

Геометрично це правило ілюструє наступний рисунок.

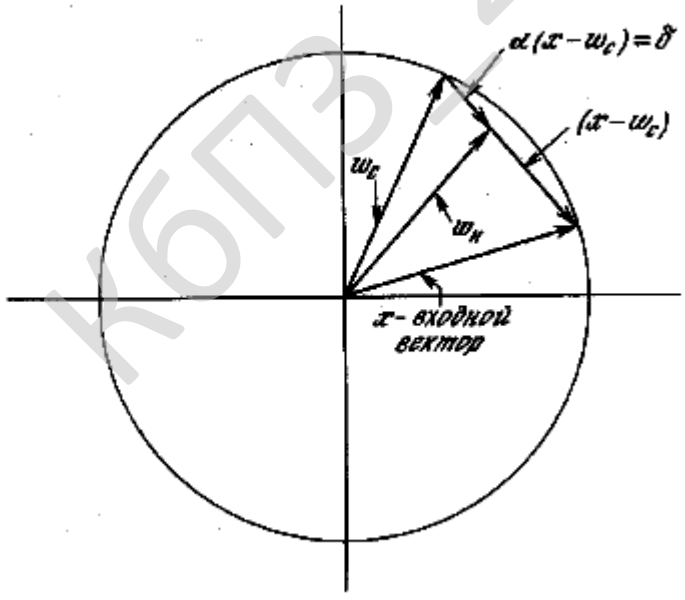

Рисунок 3.8– Корекція ваг нейрона Кохонена

Так як вхідний вектор x нормований, тобто розташований на гіперсфері одиничного радіуса в просторі ваг, то при корекції ваг за цим правилом

відбувається поворот вектора ваг убік закритого тексту. Поступове зменшення швидкості повороту α дозволяє зробити статистичне усереднення вхідних векторів, на які реагує даний нейрон. Проблема: вибір початкових значень ваг. Так як наприкінці навчання вектора ваг будуть розташовуватися на одиничній окружності, то на початку їх також бажано віднормувати на 1.0. У розглянутій нами моделі вектора ваг вибираються випадковим образом на окружності одиничного радіуса.

Проблема: якщо ваговий вектор виявиться далеко від області входу, він ніколи не дасть найкращої відповідності, завжди буде мати нульовий вихід, отже, не буде коректуватися й виявиться марним. Нейронів, що залишилися, може не вистачити для поділу вхідного простору на класи. Для рішення цієї проблеми пропонується багато алгоритмів, тут же застосовується правило «бажання працювати»: якщо який або нейрон довго не перебуває в активному стані, він підвищує ваги зв'язків доти, поки не стане активним і не почне піддаватися навчанню. Цей метод дозволяє також вирішити проблему тонкої класифікації: якщо утвориться група вхідних даних, розташованих близько друг до друга, із цією групою асоціюється й велика кількість нейронів Кохонена, які розбивають її на класи. Правило «бажання працювати» записується в наступній формі:

$$w_н = w_c + w_c\beta\,(1 - a), \qquad (3.10)$$

де $w_н$ – нове значення ваги,

w_c – старе значення,

β – швидкість модифікації,

a – активність нейрона. Чим менше активність нейрона, тим більше збільшуються ваги зв'язків.

Далі сигнал через матрицю ваг V надходить на шар Гроссберга тут шар спрацьовує по старому методу.

Алгоритм навчання

Вхідні дані: навчальна вибірка (набір вхідних векторів).

Вихідні дані: скоректовані зв'язки.

1. Пред'явити мережі вхідний вектор.

2. Виконувати ітерації до встановлення стабільного стану.

3. Для всіх вузлів мережі виконати корекцію зв'язків згідно (2) або (3).

4. Повторювати [1-3] для кожного вхідного вектора.

Розроблена система виявлення уразливих додатків у Cloud-сервісах є досить універсальною, не вимоглива до пам'яті й показала себе ефективніше, ніж класичні методи виявлення уразливих додатків у Cloud-сервісах. Достоїнствами даної системи є: швидкість аналізу, легкість адаптації, універсальність (існуючі алгоритми не захищені від такого виду аналізу).

Розглянемо функціональну схему розробленої системи. Вона зображена на рисунку 3.9

Як видно з рисунку система складається з наступних елементів:

– Додатки у Cloud-сервісах.

– Нейронної мережі зустрічного розподілу.

– Блоку порівняння вхідних даних з образами відомими системі.

– Блоку навчаючої вибірки.

– Уразливі додатки у Cloud-сервісах.

Спочатку на вхід системи подається навчаюча вибірка, за допомогою якої відбувається навчання нейронної мережі. Потім в програмі відкривається зашифрований текст. Нейронна мережа намагається розпізнати алгоритм шифрування та розшифрувати його без ключа, ще відбувається за допомогою порівняння вхідних даних з образами відомими системі після навчання.

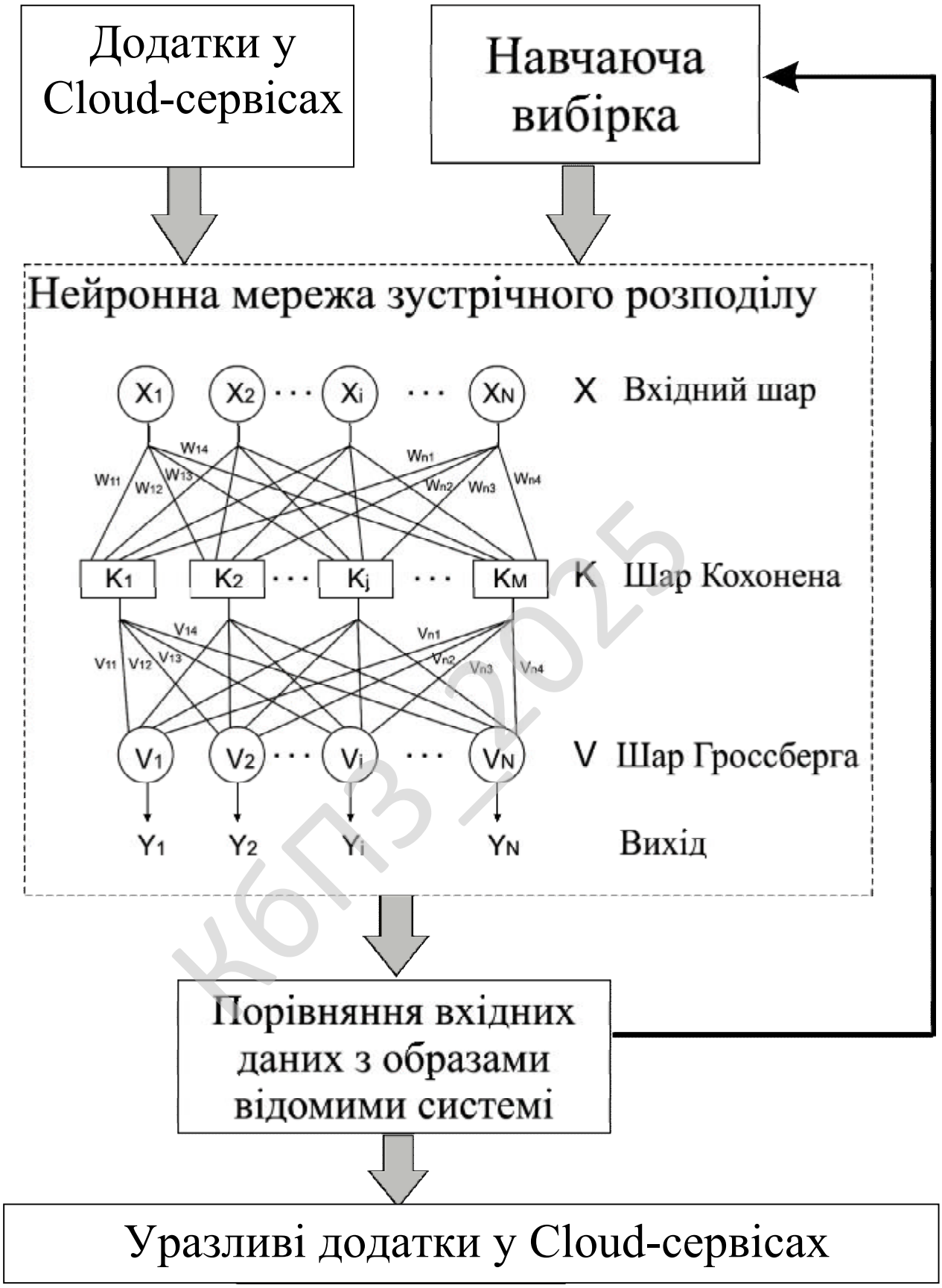

Рисунок 3.9– Функціональна схема системи

Після розшифрування тексту відбувається додавання нової інформації до бази знань. З кожним наступним використанням програма має все більше інформації для дешифрування.

У якості системи виявлення уразливих додатків у Cloud-сервісах використана нейронна мережа зустрічного розподілу, що має три шари:

1. Вхідний шар, X.
2. Шар Кохонена, K.
3. Шар Гроссберга, V.

Архітектура зустрічного розподілу вдало поєднує у собі переваги можливості узагальнення інформації мережі Кохонена й простоту навчання вихідної зірки Гроссберга. Ця архітектура ідеально підходить для швидкого моделювання систем на початкових етапах досліджень із подальшим переходом, якщо це буде потрібно, на значно більш дорожчий, але більш точний метод навчання зі зворотним поширенням помилок.

Нейронна мережа зустрічного розподілу навчається на вибірці пар векторів (X,Y) задачі подання відображення XY. Чудовою особливістю цієї мережі є здатність навчанню також і відображенню сукупності XY у себе. При цьому, завдяки узагальненню, з'являється можливість відновлення пари (XY) по одному відомому компоненту (X або Y). При пред'явлені на етапі розпізнавання тільки вектора X (з нульовим початковим Y) виконується пряме відображення – відновлюється Y, і навпаки, при відомому Y може бути відновлений відповідний йому X. Можливість рішення як прямої, так і зворотної задачі, а також гібридної задачі по відновленню окремих відсутніх компонентів робить дану нейромережну архітектуру унікальним інструментом.

Мережа зустрічного розподілу складається із двох шарів нейронів: шару Кохонена та шару Гроссберга. У режимі функціонування (розпізнавання) нейрони шару Кохонена працюють за принципом "переможець-забирає-все", визначаючи кластер, до якого належить вхідний образ. Потім вихідна зірка шару Гроссберга по сигналу нейрона-переможця в шарі Кохонена відтворює на виходах мережі

відповідний образ.

Навчання ваг шару Кохонена виконується без учителя на основі самоорганізації. Вхідний вектор спочатку нормується, зберігаючи напрямок. Після виконання однієї ітерації навчання визначається нейрон переможець, стан його порушення встановлюється рівним одиниці, і тепер можуть бути модифіковані ваги відповідної йому зірки Гроссберга. Темпи навчання нейронів Кохонена й Гроссберга повинні бути погоджені . У шарі Кохонена навчаються ваги всіх нейронів в околиці переможця, що поступово звужується до одного нейрона. Навчена нейронна мережа зустрічного розподілу може функціонувати й у режимі інтерполяції, коли в шарі Кохонена залишається не один, а декілька переможців. Тоді рівні їхньої активності пропорційно нормуються, щоб у сумі становити одиницю, а вихідний вектор визначається по сумі вихідних векторів кожної з активних зірок Гроссберга. У такий спосіб нейронна мережа робить лінійну інтерполяцію між значеннями вихідних векторів, що відповідають декільком кластерам. Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

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. Діаграма процесів розробленої системи зображена на рисунку 3.10. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

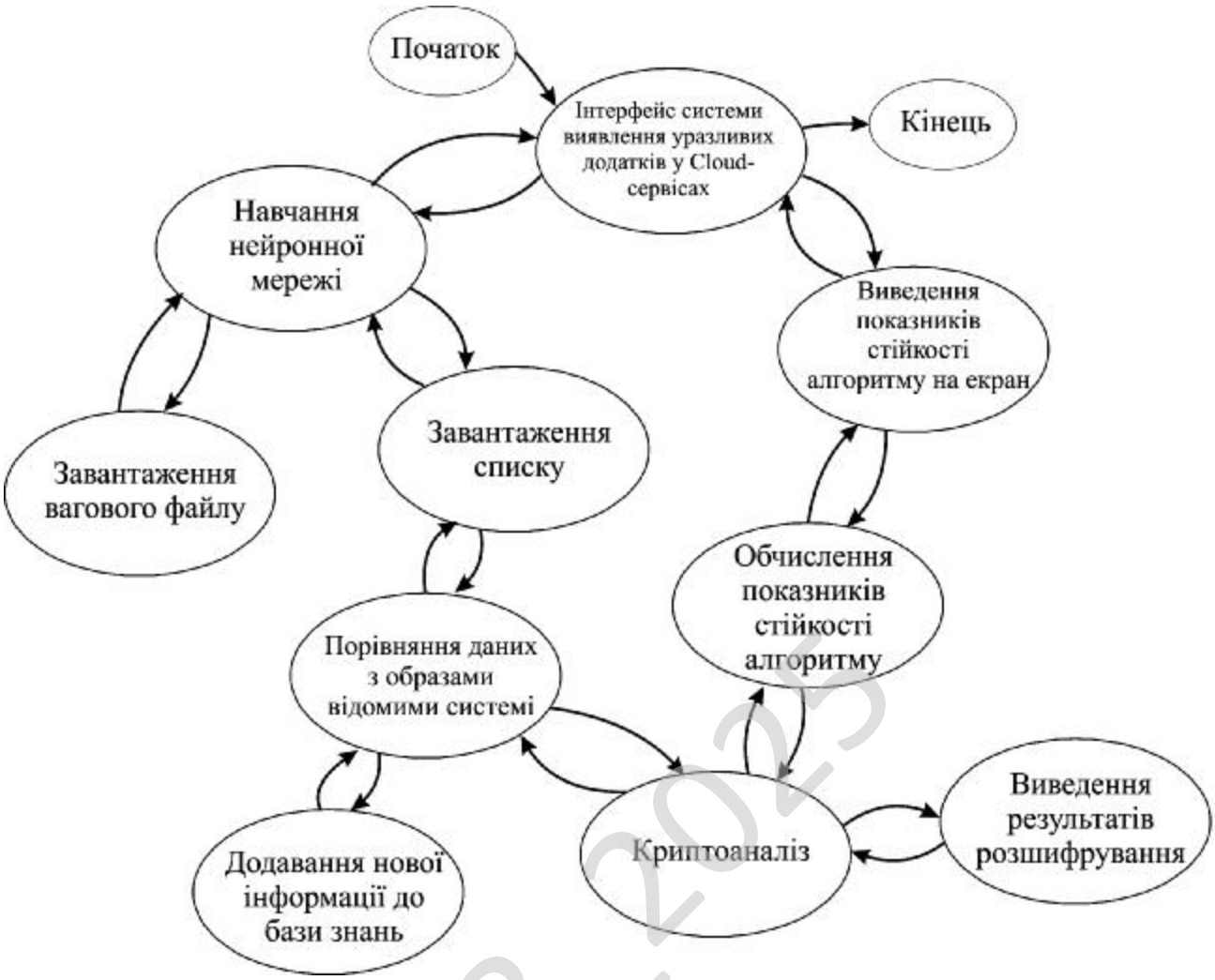

Рисунок 3.10 – Діаграма взаємодії процесів

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.

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 основою ПЗ. Тому від точності і детальності проробки 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, також те, що при розробці програми слід надати особливу увагу модулю системи виявлення уразливих додатків у Cloud-сервісах.

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.

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

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 – перевірки поточного стану та поверненням на початок схеми чи з завершенням роботи розробленого ПЗ.

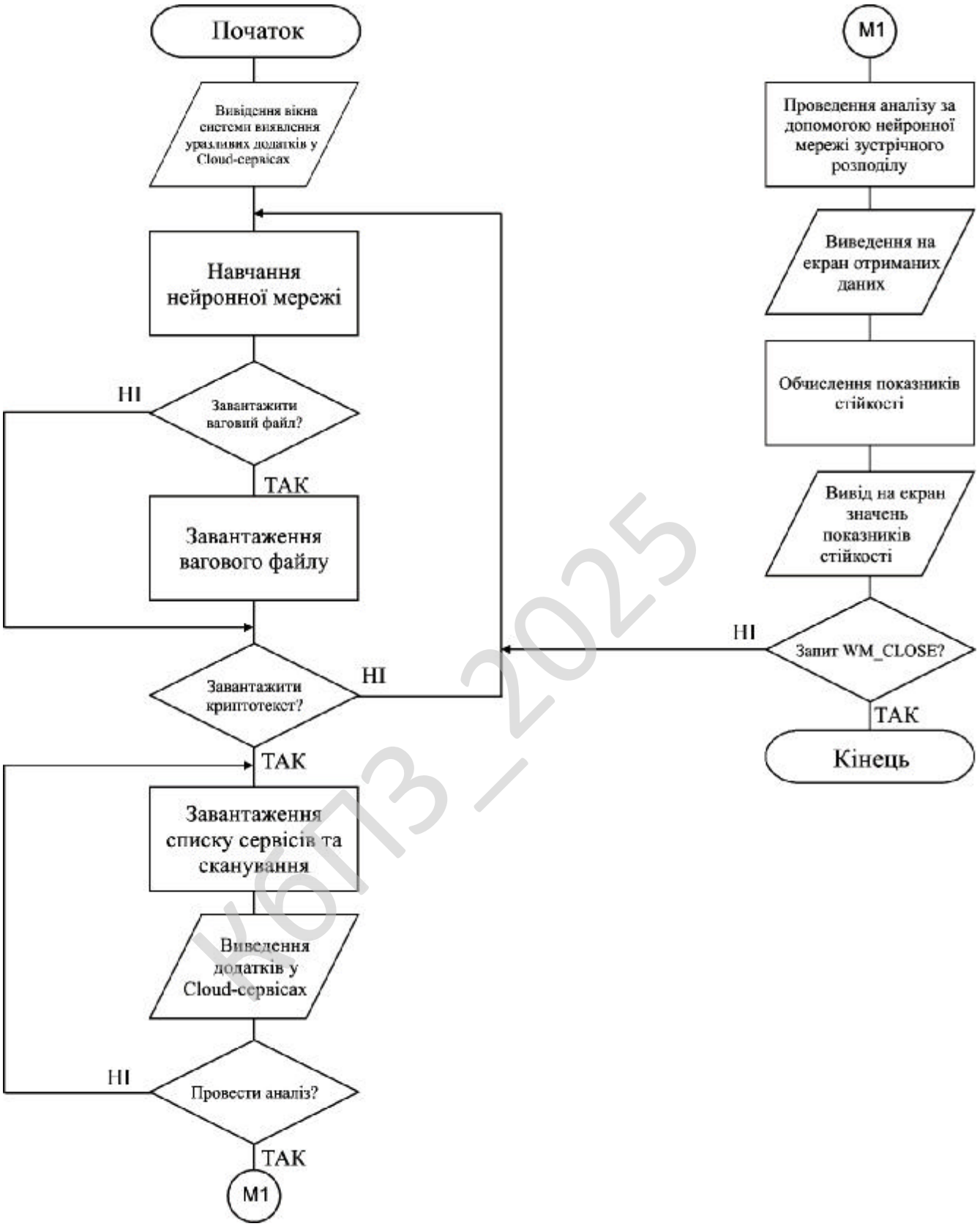

Рисунок 4.1 – Блок-схема основної програми

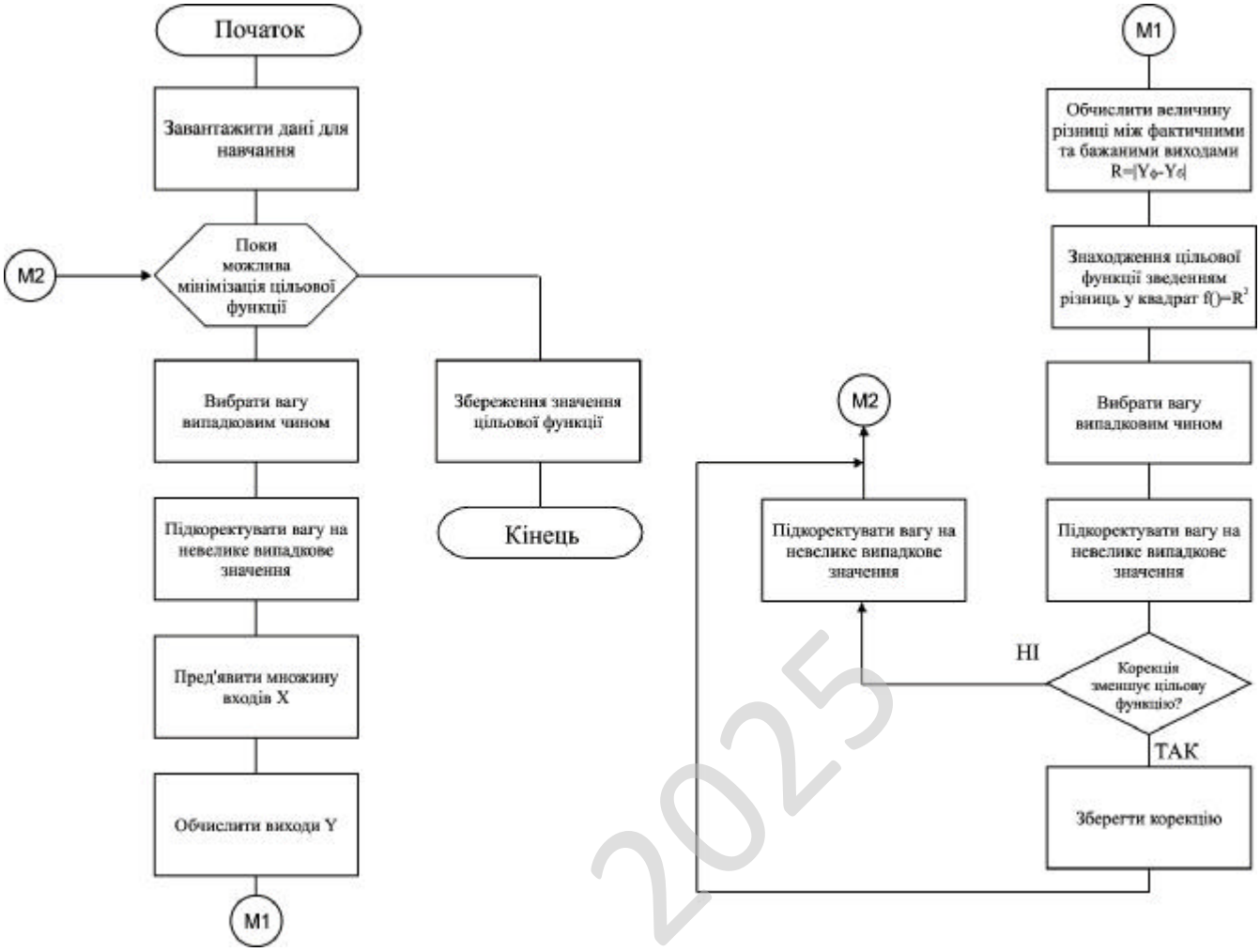

Рисунок 4.2 – Блок-схема роботи підпрограми

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

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

UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем.

UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

Розглянемо використані технології та їх основні компоненти що підтверджують правильність використаних проектних рішень. В першу чергу систему управління проектів.

Управління проектами – (Project Management) область знань з планування, організації та управління ресурсами з метою успішного досягнення цілей та завершення завдань проекту. Іноді ототожнюється з управлінням програмами, але програма – це фактично вищий рівень: група пов'язаних та взаємозалежних проектів.

Проект – це обмежений часовими рамками процес, що має визначений початок та кінець, зазвичай обмежений датою, але також може обмежуватися фінансуванням або досягненням результатів, який здійснюється для реалізації унікальних цілей та завдань, зазвичай, щоб призвести до вигідних змін або створення доданої вартості.

Тимчасова природа проектів контрастує з бізнесом (процесами), які є повторюваною, постійною або частково постійною діяльністю з виробництва продуктів або послуг. На практиці, управління вищезазначеними двома системами часто різниться і таким чином вимагає розвитку окремих технічних навичок та використання розподіленого управління ними.

Головним завданням проектного управління є досягнення всіх цілей та виконання завдань проекту, одночасно виконуючи зобов'язання щодо наперед визначених обмежень проекту.

Типовими обмеженнями є межі та зміст проекту, час, бюджет. Другорядним завданням, але амбіційнішим, є оптимізація, розподілення та інтеграція завдань, необхідних для досягнення наперед визначених цілей.

Існує певна кількість методів управління проектними активностями, включаючи Еджайл (Agile), інтерактивні, послідовні та методи розподілу на етапи.

Незважаючи на метод, що використовується, необхідно дуже уважно розглядати загальні цілі проекту, календарний план, вартість (витрати), одночасно з ролями та відповідальністю усіх виконавців та зацікавлених сторін.

Традиційний метод

Традиційний метод поділу на етапи передбачає визначення послідовності дій, що мають бути завершені. В «традиційному методі» можливо визначити 5 складових проекту (4 етапи та контроль) розвитку проекту:

1. Ініціювання.
2. Планування та розробка.
3. Виконання та впровадження.
4. Моніторинг та контроль.
5. Завершення.

Не всі проекти проходять кожен з етапів, так як проект може бути припинений до його завершення. Деякі проекти не мають етапів структурованого планування та/або моніторингу. Деякі проекти проходять стадії 2, 3 і 4 декілька разів.

Багато галузей використовують варіації зазначених етапів. Наприклад, будівельні проекти зазвичай проходять через приблизно такі етапи: Попереднє планування, Концептуальне проектування, Схематичне проектування, Розробка проекту, Будівельні креслення (або Договірні документи) та Управління будівництвом.

В розробці програмного забезпечення цей підхід відомий під назвою, дослівно, «модель водоспаду», (waterfall model), наприклад, друга група завдань виконується після першої в лінійній послідовності. Для «моделі водоспаду» використовуватимемо назву «послідовна модель».

З метою адаптації послідовної моделі при розробці програмного забезпечення багато організацій використовують методологію Раціональних уніфікованих процесів (Rational Unified Process – RUP). RUP не вимагає та однозначно не вказує на необхідність використання послідовної моделі.

Використання послідовної моделі управління проектами ефективне для невеликих, визначених проектів, але для більш великих, невизначених та нових проектів зазначена модель часто призводить до негативних результатів. «Конус невизначеності» (Cone of Uncertainty) пояснює таке явище тим, що планування, яке виконується на початкових етапах проекту є не ефективним через значний ступінь невизначеності. Це особливо актуально для розробки програмного забезпечення, оскільки така розробка часто є новим продуктом.

В проектах, де вимоги не були завершені і можуть змінюватися, використовується управління вимогами з метою розробки точного і повного визначення поведінки програмного забезпечення, що може бути базисом для його розробки. Тоді як визначення можуть змінюватися в залежності від галузі, фактичні етапи зазвичай відповідають загальним крокам вирішення проблем (problem solving) – «ідентифікація проблеми, оцінювання варіантів вирішення, вибір шляху вирішення, впровадження та оцінювання».

Критичний шлях управління проектом

Критичний шлях управління проектом (Critical Chain Project Management, далі CCPM) – це метод планування та управління проектами, який на перше місце ставить управління ресурсами (фізичними та людськими), необхідними для виконання завдань проекту. Фактично це доповнення Теорії обмежень (Theory of Constraints – TOC) для проектів. Головним завданням є підвищення продуктивності (або збільшення відсотку завершених завдань) проектів в організації. Застосовуючи перші три з п'яти основних кроків TOC, системні обмеження для всіх проектів визначаються як ресурси.

Щоб використовувати обмеження, завдання на критичному шляху отримують пріоритет вищий ніж інші активності. Загалом, проекти плануються та управляються таким чином, щоб ресурси були доступні, коли завдання критичного шляху мають розпочатися, підпорядковуючи усі інші ресурси завданням критичного шляху.

Незважаючи на тип проекту, план проекту має визначати розподілення ресурсів на рівні (Resource Leveling). Найдовша послідовність ресурсно-обмежених завдань має бути визначена, як критичний шлях. В середовищах, що мають декілька проектів, розподілення ресурсів на рівні використовується в усіх проектах. Часто, досить визначити (чи просто обрати) один наскрізний ресурс – ресурс, що виступає як обмеження в усіх проектах та послідовно розташувати проекти відповідно до доступності цього ресурсу.

Екстремальне управління проектами

В критичних оглядах Управління проектами зазначалося, що декілька методів управління проектами, які базуються на методиці Програми оцінки та контролю (Program Evaluation and Review Technique – PERT), не в повній мірі відповідають мульти-проектному середовищу сучасних компаній. Більшість з таких компаній орієнтовані на масштабні, єдино разові, не повторювані проекти, в яких усі види управління використовують інструменти проектного управління.

Використання проектної моделі для «проектів», чи скоріше «завдань», що тривають декілька тижнів, на практиці призводить до непотрібних витрат та слабкої гнучкості.

Замість використання класичного управління проектами, фахівці з управління проектами намагаються знайти різні «полегшені» методи (моделі), такі як методологія управління проектами Еджайл (Agile Project Management – дослівно «швидке, рухливе» управління проектами), включаючи Екстремальне програмування (Extreme programming) для розробки програмного забезпечення, а також техніки Скрам (Scrum – дослівно, натовп, скупчення).

Узагальнення Екстремального програмування для застосування в інших видах проектів отримало назву Екстремальне проектне управління, що може бути використане разом з Побудовою процесів (Process modeling) та принципами управління взаємодією людьми (Human interaction management).

Управління послідовністю подій

Управління послідовністю подій (Event chain methodology) – це ще один метод, який доповнює методи критичного шляху (Critical Path Method – CPM) та метод управління критичним шляхом (Critical Chain Project Management – CCPM).

Метод управління послідовністю подій – це техніка управління невизначеністю та аналізу структури і плану виконання робіт (завдань), що сфокусована на визначенні та управлінні подіями та послідовностями подій, які впливають на план реалізації проекту.

Управління послідовністю подій допомагає зменшувати негативний вплив досвіду взаємодії та особистих якостей, одночасно допомагаючи моделювати невизначеності в планах виконання проектів. Управління послідовністю подій базується на наступних принципах:

1. Ймовірний момент ризику: Активність (завдання) в більшості реальних процесів не є тривалим безперервним процесом. На завдання впливають зовнішні події, які можуть виникнути на одному з етапів посередині виконання завдання.

2. Послідовність подій: Події можуть викликати інші події, що будуть створювати послідовності подій. Такі послідовності подій можуть суттєво впливати на шлях проекту. Кількісний аналіз використовується для визначення кумулятивного ефекту таких послідовностей подій на план виконання проекту.

3. Критичні події або послідовності подій: Одиночні події або послідовності подій, що найбільш ймовірно зможуть вплинути на проект вважаються «критичними подіями» або «критичними послідовностями подій». Вони можуть бути визначені шляхом аналізу.

4. Письмове відображення проекту разом з подіями: Навіть якщо проект частково завершений і тривалість проекту, вартість, а також інформація про події, що сталися, вже відомі, існує можливість уточнення інформації про майбутні можливі події, що дозволяє спрогнозувати ефективність майбутнього виконання проекту.

5. Відображення послідовності подій: Події та послідовності подій можуть бути відображені, використовуючи діаграми послідовності подій на діаграмі Ганта.

Проекти в контрольованому середовищі

Проекти в контрольованому середовищі (Projects in controlled environments – PRINCE) – це структурований підхід до управління проектами, який був створений в 1996 році, як типовий метод управління проектами. Фактично це комбінація методології PROMPT (що еволюціонувала в методологію PRINCE) з методологію IBM MITP (Managing the implementation of the total project – MITP) – Управління впровадженням усього проекту. PRINCE2 пропонує метод управління проектами в рамках чітко визначеної структури організації. PRINCE2 описує процедури координації людей та активностей в проекті, як розробляти та контролювати проект та що робити, якщо необхідно внести зміни до проекту у зв'язку з відхиленням від плану впровадження.

Кожен процес визначено з ключовими вхідними та вихідними даними, а також цілями та активностями, які необхідно виконати для досягнення таких цілей. Це дозволяє автоматично контролювати будь-яке відхилення від плану. Розподілення на етапи, якими можливо управляти, забезпечує ефективний контроль ресурсів. Впровадження проекту відбувається структуровано та контрольовано, завдяки інтегрованому контролю за виконанням.

PRINCE2 надає єдину термінологію усім учасникам проекту. Різноманітні ролі управління та сфери відповідальності, що задіяні в проекті, повністю описані та можуть бути адаптовані, щоб відповідати складності проекту та можливостям організації.

Процесне управління

Подальший розвиток концепції контролю в проектному управлінні призводить до об'єднання методик процесного управління (Proces-based management). Ця сфера розвивається завдяки використанню Моделей зрілості (Maturity models), таких як CMMI (Capability Maturity Model Integration) –

дослівно, Інтеграція моделі можливостей зрілості) та ISO/IEC15504 (Software Process Improvement and Capability Estimation – SPICE), дослівно, Покращення процесу розробки програмного забезпечення та оцінки можливостей).

Підходи управління проектами Еджайл (Agile project management) базуються на принципах управління взаємодією людей (Human interaction management), що засновані на процесному підході до співпраці людей. Цей підхід дуже сильно відрізняється від традиційного. Розробка програмного забезпечення Еджайл чи Гнучка розробка продукту (Flexible product development) розглядають проект, як послідовність невеликих завдань, які виникають та виконуються ситуативно, відповідно до вимог обставин. Ініціація та виконання є скоріше адаптивними до зовнішніх обставин, ніж завчасно повністю спланованим процесом.

Групи процесів управління проектами

Традиційно, управління проектами включає наступний перелік елементів: чотири групи процесів та систему контролю. Незалежно від методології чи термінології, що застосовується, використовуються одні й ті самі базові процеси управління проектами.

Групи процесів зазвичай включають:

– Ініціювання.

– Планування чи розробка.

– Експлуатація чи виконання.

– Моніторинг і контроль.

– Завершення.

У проектному середовищі з дослідницьким нахилом (наприклад, Дослідження та проектування, (Researsh and Development)) зазначені етапи можуть доповнюватися точками прийняття рішень, де вирішується чи потрібно продовжувати чи припинити впровадження проекту.

Ініціювання

Процес ініціювання визначає зміст та межі проекту. Якщо зазначений етап виконується не в повній мірі, то здебільшого проект не буде відповідати вимогам бізнеса. Ключовими контрольними індикаторами для зазначеного етапу є розуміння бізнес середовища та включення усіх необхідних контрольних індикаторів в проект. Усі невідповідності, проблемні питання мають бути розглянуті та рішення щодо таких питань мають бути прийняті з метою їх усунення.

Етап ініціювання має містити план, що висвітлює такі питання:

– Аналіз бізнес-потреб/вимог в вимірюваних показниках.

– Огляд існуючих процесів.

– Фінансовий аналіз витрат і доходів, включаючи бюджет.

– Аналіз зацікавленими сторонами, включаючи користувачів, працівників підтримки проекту.

– Устав проекту, включаючи витрати, завдання, результати та календарний план впровадження.

Планування та розробка

Після етапу ініціювання, проект планується з необхідним рівнем деталізації. Головне завдання – спланувати час, витрати та ресурси з метою адекватної оцінки роботи, яку необхідно виконати, та ефективного управління ризиками протягом впровадження проекту.

Аналогічно групі процесів ініціювання, недостатньо розроблений план значно знижує шанси проекту успішно завершити поставлені перед ним завдання.

Планування проекту загалом складається з:

– Визначення «як планувати?» (наприклад, за рівнем деталізації чи етапами впровадження).

– Розробка документу, що визначає зміст та межі проекту.

– Визначення групи відповідальних, що плануватимуть проект.

– Визначення результатів проекту та створення структури декомпозиції робіт.

– Визначення завдань, які необхідно виконати для досягнення цілей проекту, та логічне послідовне об'єднання таких завдань.

– Оцінка вимог до ресурсів для забезпечення виконання завдань.

– Оцінка часу та витрат на виконання завдань.

– Розробка календарного плану-графіку.

– Розробка бюджету.

– Планування ризиків.

– Отримання формального погодження початку роботи.

Додаткові процеси, такі як планування комунікацій, визначення ролей та рівня відповідальності, закупівель в рамках проекту, а також проведення попередньої зустрічі учасників проекту (Kick-off meeting) загалом є рекомендованими.

Для проектів з розробки нових продуктів, концептуальна розробка продукту може проводитися одночасно з активностями з планування, які можуть допомогти групі з планування, шляхом інформування про визначені результати проекту та заплановані завдання.

Виконання

Виконання складається з процесів, необхідних для завершення робіт, визначених в плані проекту, з метою виконання вимог проекту. Процес виконання включає координування людей та ресурсів, одночасно з інтеграцією та виконанням завдань проекту відповідно до плану управління проектом. Результати проекту є результатами виконання завдань проекту відповідно до плану проекту.

Моніторинг та контроль

Моніторинг та Контроль складається з процесів, що виконуються з метою огляду стану виконання проекту, щоб потенційні проблеми були визначені вчасно і коригуючи дії можливо було виконати, з метою контролю виконання проекту.

Головним здобутком є регулярний огляд та оцінювання процесу виконання проекту з метою визначення відхилень від плану управління проектом.

Моніторинг та контроль включає:

– Вимірювання поточного виконання завдань проекту (де ми є зараз?).

– Моніторинг змінних складових проекту (зміст та межі проекту, витрати тощо) в порівнянні до плану управління проекту та базового плану виконання проекту.

– Визначення коригуючих дій, з метою правильного вирішення відкритих питань та ризиків (Як ми можемо привести фактичний стан виконання до планового виконання?).

– Вплив на фактори, що можуть призвести до порушення інтегрованого контролю змін, для того щоб лише погоджені зміни впроваджувалися.

– В багатоетапних проектах процеси моніторингу та контролю забезпечують зворотний зв'язок між етапами проекту, задля забезпечення коригуючих чи превентивних дій з метою приведення проекту у відповідність з планом управління проектом.

Підтримка проекту – це постійний процес, що включає:

– Постійну підтримку кінцевих користувачів.

– Виправлення помилок.

– Оновлення програмного забезпечення.

На цьому етапі, аудитори мають звертати увагу на ефективність та швидкість вирішення проблем користувачів.

Наприклад, протягом впровадження будь-якого будівельного проекту зміст та межі проекту можуть змінюватися. Зміни – це нормальна та очікувана частина процесу будівництва. Зміни можуть бути результатом необхідних модифікацій конструкції, відмінностях будівельного майданчика, доступності матеріалів, змін за вимогою підрядної організації, впливом третіх сторін і т. д.

Окрім впровадження змін, вони мають бути закріплені в документах, щоб відобразити кінцевий результат. Описаний процес отримав назву Управління

змінами (Change management). Зазвичай власник вимагає остаточне документування змін, щоб зафіксувати усі зміни, а іноді, будь-яку зміну, що змінює матеріальні пакети завершеної роботи.

Документування здійснюється за допомогою договірних документів, зазвичай, але не обов'язково, обмежуючись кресленнями. Кінцевий продукт такого процесу отримав назву, дослівно, «креслення, як побудовано» (As-built drawing), чи більш спрощено, «Як побудовано» (As built).

Вимога щодо надання таких креслень є нормою будівельних договорів. Коли зміни внесені до проекту, життєздатність проекту має бути оцінена ще раз. Важливо не втратити початкових цілей та завдань проекту. Коли усі зміни зібрані, прогнозований результат може не виправдати інвестиції у проект.

Завершення

Завершення включає формальне прийняття проекту та відповідно його закриття. Проводяться адміністративні активності, включаючи передачу до архіву робочої документації та документування здобутого досвіду.

Цей етап складається з:

– Закриття проекту: Завершення усіх активностей у всіх групах процесів з метою формального закриття проекту або етапу проекту.

– Закриття договору (контракту): Завершення та оплата кожного договору, включаючи вирішення будь-яких відкритих питань, та закриття кожного договору, що відноситься до проекту в цілому або до певного етапу проекту.

Системи контролю проектів

Контроль проекту – це елемент, який забезпечує відповідність проекту графіку виконання та бюджету. Контроль проекту починається з планування та закінчується звітом з виконання проекту, пронизуючи кожен елемент процесу управління проектом. Кожен проект має бути оцінений щодо рівня необхідного контролю: забагато контролю означає втрату часу, замало контролю означає збільшення ризиків. Якщо контроль проекту впроваджений не вірно, вартість для

бізнесу пояснюється у термінах помилок, виправлень та додаткових витрат на аудит.

Системи контролю необхідні для витрат, ризиків, якості, комунікацій, часу, змін, закупівель та людських ресурсів. До того ж аудитори мають визначити наскільки проекти впливають на фінансову звітність, наскільки достовірну інформацію отримують замовники і скільки точок контролю існує. Аудитори мають розглянути процес розробки та процедури на предмет способу впровадження. Процес виробництва та якість кінцевого продукту також може бути оцінений, якщо така потреба виникає. Бізнес може забажати від аудиторської фірми фіксування проблем на ранніх етапах з метою зменшення зусиль необхідних на виправлення. Аудитор може виступати як консультант з контролю, частина проектної команди або як окремий аудитор, частина аудиту. Бізнес іноді використовує формалізовані процеси розробки систем. Це допомагає підтвердити успішність розробки.

Формальний процес ефективніший у створенні сильних точок контролю. Аудитори мають перевірити такий процес та підтвердити його якісну організацію та відповідність практиці. Гарний формальний план впровадження системи характеризує:

– Стратегія, задля приведення розробки до загальніших цілей організації.

– Стандарти для нових систем.

– Політики управління проектом щодо часу та бюджету.

– Процедури, що описують процес.

– Оцінка якості змін.

4.2 Захист розробленого програмного забезпечення

Розроблене програмне забезпечення захистимо за допомогою наступного алгоритму захисту інформації DSA.

Відправник і одержувач електронного документа використовують при

обчисленні великі цілі числа: G і P – прості числа, L біт кожне ($512 < L < 1024$); q – простої число довжиною 160 біт (дільник числа (P-1). Числа G, P, q є відкритими й можуть бути загальними для всіх користувачів мережі.

Відправник вибирає випадкове ціле число X, $1 < X < q$. Число X є секретним ключем відправника для формування електронного цифрового підпису. Потім відправник обчислює значення $Y = G^X \bmod P$. Число Y є відкритим ключем для перевірки підпису відправника. Число Y передається всім одержувачам документів.

Цей алгоритм також передбачає використання однобічної функції гешування h(-). У стандарті DSS визначений алгоритм безпечного гешування SHA. Для того щоб підписати документ M, відправник гешує його в ціле геш-значення m: $m = h(M)$, $1 < m < q$, потім генерує випадкове ціле число K, $1 < K < q$, і обчислює число r: $r = (G^K \bmod P) \bmod q$. Потім відправник обчислює за допомогою секретного ключа X ціле число s:

$$s = \frac{m + r * X}{K} \bmod q \ .$$

Пара чисел r і s утворить цифровий підпис $S = (r, s)$ під документом M. Таким чином, підписане повідомлення являє собою трійку чисел $[M,r,s]$. Одержувач підписаного повідомлення $[M, r, s]$ перевіряє виконання умов $0 < r < q$, $0 < s < q$ і відкидає підпис, якщо хоча б одна із цих умов не виконана. Потім одержувач обчислює значення $w = 1/s \bmod q$, геш-значення $m = h(M)$ і числа $u_1 = (m*w) \bmod q$, $u_2 = (r*w) \bmod q$. Далі одержувач за допомогою відкритого ключа Y обчислює значення $v = ((G^{U1} * Y^{U2}) \bmod P) \bmod q$ і перевіряє виконання умови $v = r$. Якщо умова $v = r$ виконується, тоді підпис $S = (r,s)$ під документом M визнається одержувачем справжнім.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ системи виявлення уразливих додатків у Cloud-сервісах яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Навігаційне меню: Файл; Дані; Cloud параметри; Налаштування; Довідка.

– Вікно виведення результату роботи системи.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

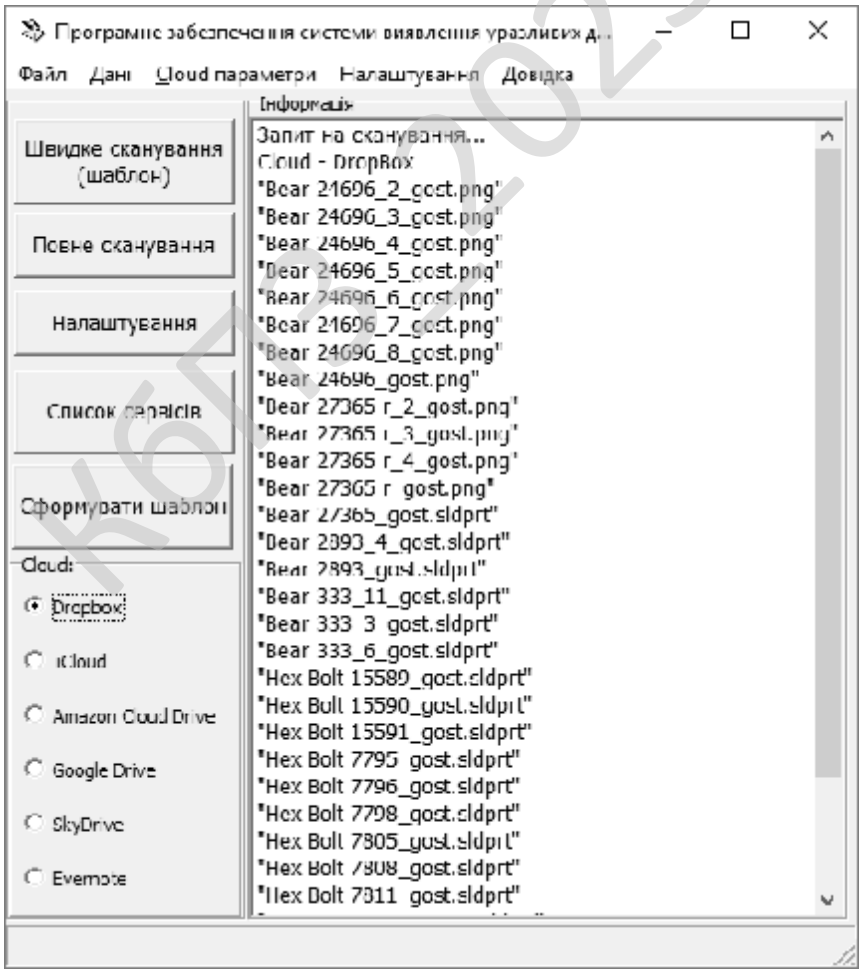

Рисунок 5.1 – Головне вікно ПЗ

Надаючи клієнтам API або просто ресурси для хмарного хостингу, його власники повинні бути впевнені в тім, що розміщений у хмарі код не знизить загальну захищеність. Поки ж у більшості хмарних сервісів клієнтські додатки приймаються «як є», а кількість атак на уразливі додатки тільки росте. Як розв'язати цю проблему? Проводити пентести? Організовувати незалежне приймання додатків, як це робиться при випуску тиражируємих продуктів?

Але чи погодяться клієнти хмарних сервісів на такі міри? Чи будуть вони оплачувати дослідження своїх додатків, розміщених у хмарному сервісі? Навряд чи. Здебільшого це бізнесмени, і швидкість запуску сервісів для них важливіше, ніж їхня захищеність. Ресурси, у тому числі людські, у них теж обмежені, і вибір між «додати нові функції» і «виправляти уразливості в старих» для них теж очевидний – не виправляти. Їм немає справи до ризиків інших клієнтів, як і до ризиків, що виходять від інших клієнтів сервісу, – вони підписали SLA з конкретним сервісом і очікують, що всі проблеми із захищеністю будуть вирішуватися провайдером.

Швидше за все, оплачувати такі процедури прийде власникові хмарного сервісу, оскільки уразливість розміщених програм прямо впливає на захищеність сервісу й дотримання SLA. Деякі центри обробки даних, де розміщаються клієнтські додатки, уже вводять ті або інші елементи процесу приймання. Думаю, не за горами й повноцінними процедурами, знайомі по прийманню у фонди алгоритмів і програм.

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

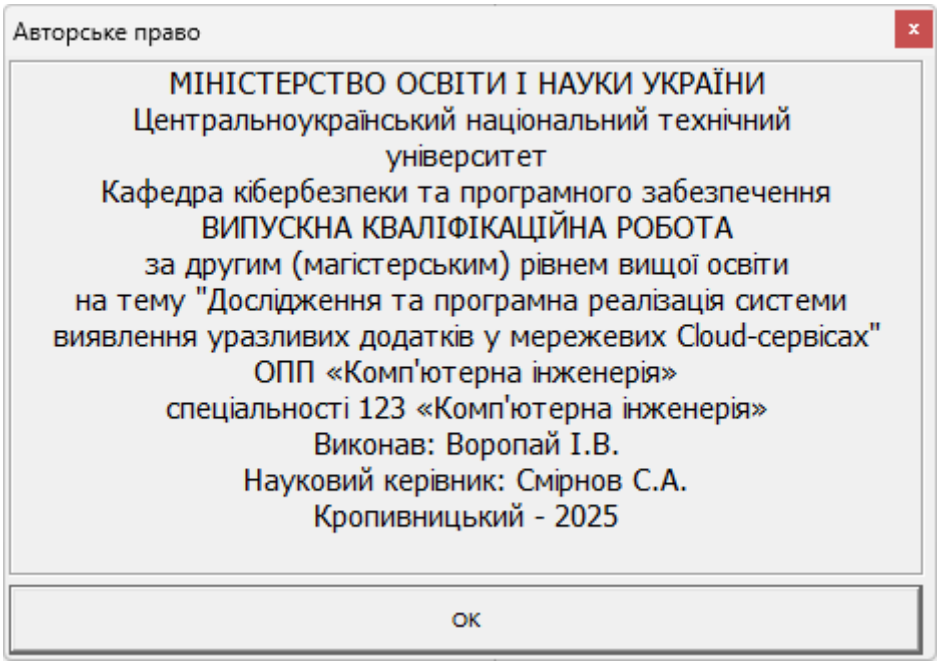

Рисунок 5.2 – Авторське право

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 чорної скриньки.

Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ITc).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його OT).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ITc;

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору OT.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій.

– Помилок інтерфейсу.

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних.

– Помилок характеристик (необхідна ємність пам'яті і т.д.).

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – Shareware.

Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми. В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання. Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно. Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (CD диск, флешку, ключ). Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості. Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (зареєструватися), заплативши авторові певну суму. В іншому випадку користувач повинен припинити використання ПЗ та видалити його зі свого комп'ютера.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи виявлення уразливих додатків у мережевих Cloud-сервісах.

Метою розробки є дослідження та програмна реалізація системи виявлення уразливих додатків у мережевих Cloud-сервісах.

Об'єктом дослідження є процес виявлення уразливих додатків у мережевих Cloud-сервісах.

Предметом дослідження є методи виявлення уразливих додатків у мережевих Cloud-сервісах.

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

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

– Удосконалено метод виявлення уразливих додатків у мережевих Cloud-сервісах.

– Розроблено вітчизняний продукт виявлення уразливих додатків у мережевих Cloud-сервісах, який має більш широкі можливості, на відміну від існуючих аналогів.

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ІТ-ПРОЄКТУ

7.1 Визначення цільової аудиторії кінцевого готового продукту

Результати цього дослідження можуть бути особливо цікавими для компаній, що працюють у сфері кібербезпеки, адже питання виявлення уразливостей у хмарних додатках стає дедалі актуальнішим. Зі зростанням кількості сервісів, які функціонують у Cloud-середовищі, зростає і ризик витоку даних, тому для фахівців із безпеки така система може стати корисним інструментом у їхній щоденній роботі. Вона допоможе автоматизувати процеси моніторингу, зменшити навантаження на аналітиків і швидше реагувати на потенційні загрози.

Крім того, результати розробки можуть зацікавити компанії, які займаються розробленням програмного забезпечення, адже інтеграція системи виявлення уразливостей на етапі тестування дозволяє зменшити кількість помилок у готовому продукті. Освітні заклади також можуть використовувати цю систему як навчальний інструмент для студентів спеціальностей, пов’язаних із ІТ та інформаційною безпекою, демонструючи на практиці принципи пошуку та усунення вразливостей.

Для державних установ і підприємств, які працюють з великими масивами персональних або фінансових даних, така система також є цінною. Вона може стати частиною їхньої політики інформаційної безпеки, допомагаючи дотримуватись вимог законодавства у сфері захисту даних. Зрештою, користь від упровадження такої системи є універсальною – вона підвищує рівень безпеки не лише для великих компаній, а й для малого бізнесу, який часто не має власного відділу ІТ-захисту.

7.2 Оцінка привабливості шляхом застосування методів експертних оцінок

Для оцінки привабливості системи виявлення уразливих додатків у Cloud-середовищах було проведено опитування експертів із галузі інформаційної безпеки, розробників програмного забезпечення та адміністраторів хмарних платформ. Експерти оцінювали систему за такими критеріями, як точність виявлення уразливостей, швидкість аналізу, зручність інтерфейсу, рівень автоматизації та можливість інтеграції з іншими системами безпеки. Середня оцінка привабливості продукту склала 8,7 бала з 10, що свідчить про високий рівень зацікавленості.

Особливо позитивні відгуки система отримала за точність визначення ризиків і можливість налаштування глибини перевірки. Це дозволяє використовувати її як у невеликих компаніях, так і у великих корпораціях, де важливо виявляти потенційні загрози ще до того, як вони стануть критичними. Експерти також підкреслили, що завдяки автоматичній аналітиці система допомагає зменшити людський фактор, що є суттєвим для підвищення надійності процесів безпеки. Високі оцінки за масштабованість і сумісність із популярними Cloud-платформами підтвердили перспективність розробки на ринку.

7.3 Вибір методу оцінки вартості ПЗ

Для оцінки вартості програмної реалізації системи виявлення уразливих додатків доцільно використати комбінований метод, що поєднує витратний і дохідний підходи. Витратний метод дозволяє розрахувати реальні витрати на розробку, тестування, ліцензування компонентів, а також оплату праці розробників і спеціалістів із безпеки. Це забезпечує прозорість і точність розрахунків. Дохідний підхід, своєю чергою, допомагає оцінити майбутню економічну вигоду від комерційного використання системи, тобто можливий

прибуток від продажу ліцензій чи надання послуг технічної підтримки. Таке поєднання методів дозволяє не лише визначити, скільки коштує створення системи, а й оцінити, наскільки швидко вона може окупитися. Наприклад, якщо система буде впроваджена у кількох компаніях одночасно або запропонована як SaaS-рішення, то прибуток може перевищити первинні витрати вже протягом першого року. Це робить комбінований метод найбільш раціональним для сучасних Cloud-проєктів, які мають як технічну, так і комерційну складову.

7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості

Результати розрахунку зведемо до таблиці 7.1.

Таблиця 7.1 – Економічна ефективність впровадження системи

Показник	До впровадження	Після впровадження	Економічний ефект
Кількість уразливостей, що призводили до інцидентів на рік	10	3	–7
Середні втрати від одного інциденту	40 000 грн	10 000 грн	–30 000 грн
Річні витрати на аудит безпеки	150 000 грн	100 000 грн	–50 000 грн
Витрати на впровадження системи (одноразово)	—	—	300 000 грн
Річний економічний ефект	—	—	260 000 грн
Термін окупності	—	—	1,15 року

Впровадження системи дозволяє скоротити збитки від інцидентів безпеки майже у чотири рази, зменшити витрати на аудит на третину та окупити інвестиції менш ніж за півтора року. Додатковою вигодою є зростання довіри клієнтів і стабільність роботи Cloud-сервісів.

7.5 Пропозиція алгоритму просування проєкту розробки ПЗ

Просування проєкту системи виявлення уразливих додатків у Cloud-середовищах має починатися з формування репутації надійного рішення. Першим кроком є створення демонстраційної версії системи, яка дозволяє потенційним користувачам протестувати функціонал. Це сприятиме формуванню довіри. Далі варто залучити профільні IT-компанії, які займаються розробкою або адмініструванням Cloud-платформ, і запропонувати партнерські програми.

На другому етапі можна організувати публічні презентації, вебінари та участь у конференціях із кібербезпеки. Такі заходи допоможуть розповісти про переваги розробки, показати її ефективність і залучити перших клієнтів. Не менш важливим буде створення офіційного сайту з описом функціональних можливостей, порівняннями з конкурентами та відгуками користувачів. Після цього можна масштабувати просування через соціальні мережі та професійні онлайн-спільноти.

7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ

Для ефективної реалізації проєкту важливо орієнтуватися на партнерство з провайдерами хмарних послуг. Це дозволить інтегрувати систему безпосередньо у Cloud-платформи як додатковий рівень безпеки. Також можна запропонувати ліцензування за моделлю “Software as a Service”, що зробить продукт доступним навіть для малого бізнесу.

Іншим напрямом є співпраця з освітніми установами, які можуть використовувати систему як навчальний інструмент для студентів ІТ-напрямів. Окрім цього, розміщення системи в каталогах відкритого ПЗ і на маркетплейсах допоможе охопити ширшу аудиторію користувачів без значних витрат на рекламу.

7.7 Визначення ключових факторів успіху конкретного проєкту

Успіх цього проєкту визначається трьома головними факторами – точністю, надійністю та довірою. Якщо система демонструє високу ефективність виявлення уразливостей і мінімум хибних спрацьовувань, вона стає цінною для ІТ-фахівців. Не менш важливим є стабільне оновлення бази даних загроз і алгоритмів, що дозволяє системі залишатися актуальною навіть у швидкозмінному середовищі кіберзагроз.

Крім технічної частини, велику роль відіграє репутація розробників. Відкритість, підтримка користувачів і постійне вдосконалення продукту створюють основу для довгострокового успіху. Якщо система буде простою у використанні, але при цьому ефективною, вона знайде своє місце серед сучасних інструментів для захисту Cloud-інфраструктури.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Аналізуючи умови працівників ІТ-сфери, на перший погляд, може здатися, що працівники сфери інформаційних технологій не підпадають під ризи на виробництві, та якщо більш глибоко розглянути умови і специфіку праці фахівців сфері it-індустрії, можна виявити ряд факторів які будуть мати негативний влив як на стан охорони праці, так і на самого it-фахівця зокрема. Сюди можна віднести як невідповідність освітлення, так і високий рівень шуму, що негативно позначатимуться як на емоційному так і на фізичному стані фахівця, призводитимуть до зниження ефективності праці та виробничих травм. Також, важливим моментом охорони праці ІТ-фахівця є врахування його психологічних можливостей (швидкість реакції, особливості пам'яті та уваги, емоційний стан тощо). Для того, щоб забезпечити ефективну роботу it-фахівця, потрібно враховувати та максимально компенсувати такі негативні фактори як: надмірне нервово-емоційне навантаження, довготривалі статичні перевантаження, обмежена рухова активність. Всі ці чинники призводить до різноманітних відхилень у стані здоров’я, зокрема до перевтоми, зниження фізичної та розумової працездатності, неврозів, захворювань серцево-судинної системи тощо. Метою даного розділу є огляд конкретних умов праці спеціаліста у сфері it-індустрії. Завданнями для даного розділу є: аналіз умов праці на робочому місці фахівця it-індустрії, розробка конкретних рекомендацій щодо покрашення умов праці фахівців it-індустрії, огляд пожежної безпеки на ІТ-підприємстві та розрахунок системи загального штучного освітлення виробничого приміщення де працюють ІТ-фахівці.

На робочому місці ІТ-фахівця (або програміста) виникають небезпечні та шкідливі для безпечної життєдіяльності фактори:

– підвищений рівень шуму;

– несприятливі мікрокліматичні умови;

– недостатній рівень освітленості;

– шкідливі речовини;

– підвищений рівень електромагнітних випромінювань радіочастот;

– висока напруга електричної мережі;

– статична електрика та інші.

8.2 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

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

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	3
Довжина	4,6
Висота	3

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого*

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	6,9
Об'єм, V	$м^3$	не менше 20.0	20,7

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

У зазначеному приміщенні працюють двоє людей. За даними, які наведено у табл. 8.1, та табл. 8.2, можна зробити висновок, що площа та об'єм приміщення у розрахунку на одно робоче місце програміста не відповідають нормативним вимогам ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [2], але відповідають нормативним вимогам Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»). Таким чином можна зробити висновок, що санітарно-гігієнічні умови праці на робочому місці програміста відповідають вимогам.

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, робота, виконувана в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину.

Вологість повітря в приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

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

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

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

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Вологість,%	Швидкість повітря, м/с	Температура,°С	Вологість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-24	40-55	0,12
Тепла	23-25	50-70	0,1	24-25	50-65	0,9

В літню пору застосовується кондиціонер.

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

У приміщенні знаходяться наступні джерела шуму: принтер Xerox WorkCentre 3025BI (3025VBI), електродвигуни вентиляторів ЕОМ.

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

З 2019 року діють Державні будівельні норми України “Природне і штучне освітлення” – ДБН В.2.5-28:2018 [1], у яких прописані вимоги до використання всіх освітлювальних приладів, у т.ч. світлодіодних.

Працю працівника, який постійно працює за комп’ютером, згідно ДБН В.2.5-28:2018 [1], можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом

(під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 Лк. [1], Крім того, все поле зору повинно бути освітлено достатньо рівномірно – це основна гігієнічна вимога. Оскільки яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.3 Розробка заходів з умов поліпшення охорони праці

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

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

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значеннь;

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

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов’язковою наявністю масок-

клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга) [9].

Регулярна наочне знайомство персоналу із шляхами для евакуації людей із приміщення відповідно до плану евакуації, забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

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

8.4 Розрахункова частина

Питання охорони праці та правила безпеки при роботі з офісною, комп'ютерною та мережевою технікою розглянуті у працях вітчизняних вчених [11].

Запорукою безпечної роботи є виконання вимог електричної безпеки, оскільки все офісне обладнання заживлюється від електричної мережі. Одним з необхідних засобів електричної безпеки є встановлення захисного заземлення. Початкові дані, необхідні для розрахунку захисного заземлення:

– допустимий опір розповсюдженню струму в землі від заземлю-вального пристрою $R_{зн}$ = 10 Ом;

– питомий опір грунту в місці встановлення заземлювача $\rho_з$ = 100 Ом/м;

– тип грунту – суглинок;

– тип заземлювача – труба, діаметром d=0.045 м і довжиною l =2.2 м; – конструкція заземлювача – розташування заземлювачів по контуру. Розрахунок проводимо за стандартною методикою [7].

Визначимо розрахунковий опір землі:

$$\rho_{рз} = \phi \cdot \rho_3$$

де ϕ – коефіцієнт сезонності, що враховує коливання питомого опору при зміні

вологості ґрунту протягом року; при використанні заземлювача довжиною l = 2.2 м при глибині закладання від вершини h = 0.6 м ϕ = 1.1 для четвертої кліматичної зони.

Схема розташування заземлювачів показана на рисунку 8.1.

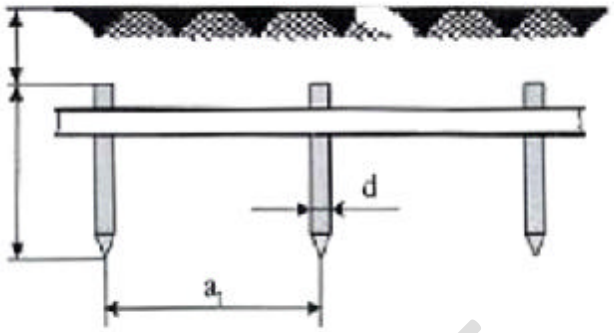

Рисунок 8.1 – Схема розташування заземлювачів

Опір землі:

$$\rho_{рз} = 1{,}1 \cdot 100 = 110 \text{ Ом·м}$$

Опір Rв, розповсюдженню струму в землі від одного вертикального заземлювача:

$$R_B = \frac{\rho_{рз}}{2\pi \cdot l}\left(\ln\frac{2 \cdot l}{d} + 0.5\ln\frac{4t + l}{4t - l}\right)$$

де

l – довжина заземлювача (l = 2.2 м);

d = 0.045 м – діаметр заземлювача при U<1 кВ та при S<100 кВА;

t – відстань від поверхні до середини заземлювача:

$$t = h + l/2 = 0.6 + 2.2/2 = 1.7 \text{ м.}$$

$$R_в = \frac{110}{2 \cdot 3.14 \cdot 2.2}\left(\ln\left(\frac{2 \cdot 2.2}{0.045}\right) + 0.5 \cdot \ln\left(\frac{4 \cdot 1.7 + 2.2}{4 \cdot 1.7 - 2.2}\right)\right) = 38.9 \text{ Ом}$$

Визначаємо потрібну кількість заземлювачів:

$$n' = \frac{R_в}{R_{зн}} = \frac{38.9}{10} = 3.9 \approx 4 \text{ шт.}$$

Коефіцієнт використання вертикальних заземлювачів враховує ефект екранування. При вибраному значенні k = a/l, де a – відстань між вертикальними заземлювачами, м; k = 1 при a = 1.8 м. Коефіцієнт використання вертикального заземлювача за довідковими даними дорівнює $\eta_В = 0,6$.

Кількість вертикальних заземлювачів з урахуванням коефіцієнту використання $\eta_В$ приблизно складає

$$n = \frac{R_в}{R_{зн} \cdot \eta_в} = \frac{38.9}{10 \cdot 0.6} = 6.48 \approx 7 \text{ шт.}$$

Довжина горизонтального заземлювача, необхідна для розміщення вертикальних заземлювачів по контуру

$$L = a \cdot n = 1.8 \cdot 7 = 12.6 \text{ м}$$

Опір горизонтального заземлювача $R_Г$, Ом, прокладеного на глибині h = 0.6 м від поверхні землі буде

$$R_г = \frac{R_{рз}}{2 \cdot 3.14 \cdot L} \cdot \ln\frac{2 \cdot L^2}{b \cdot h} = \frac{110}{2 \cdot 3.14 \cdot 12.6} \cdot \ln\frac{2 \cdot 12.6^2}{0.06 \cdot 0.6} = 13.57 \text{ Ом}$$

де b = 0.06 м – ширина сталевої смуги, з якої виготовлений заземлювач.

Обчислюємо загальний опір:

$$R_з = \frac{R_в \cdot R_г}{n \cdot R_г \cdot \eta_в + R_в \cdot \eta_г} = \frac{38.9 \cdot 13.57}{6 \cdot 13.57 \cdot 0.6 + 38.9 \cdot 0.34} = 7.93 \text{ Ом}$$

де $\eta_Г$ – коефіцієнт використання горизонтального заземлювача ($\eta_Г$ = 0.34).

Маємо 7.93 < 10 Ом ((за потужності генераторів та трансформаторів 100 кВт і менше)), отже нормативне обмеження $R_з < R_{з.норм}$ виконується.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи виявлення уразливих додатків у мережевих Cloud-сервісах.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів виявлення уразливих додатків у мережевих Cloud-сервісах.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем виявлення уразливих додатків у мережевих Cloud-сервісах.

– Досліджена система виявлення уразливих додатків у мережевих Cloud-сервісах.

– На основі отриманих результатів досліджень створена програмна реалізація системи виявлення уразливих додатків у мережевих Cloud-сервісах.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання виявлення уразливих додатків у мережевих Cloud-сервісах.

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

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

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

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

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм DSA.

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.

Проведено маркетингове та економічне обґрунтування ІТ-проєкту, що дозволило визначити ключові фактори успіху даного проекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Воропай І.В. Дослідження та програмна реалізація системи виявлення уразливих додатків у мережевих Cloud-сервісах // Збірник праць молодих науковців ЦНТУ. – Вип. 15. – Кропивницький: ЦНТУ, 2025.

2. Vijay Kumar Velu. Mastering Kali Linux for Advanced Penetration Testing Packt Publishing Ltd. 2022. 573 p.

3. Josh Armitage. Cloud Native Security Cookbook. O'Reilly Media. 2022. 516 p.

4. Massimo Bertaccini. Cryptography Algorithms. Packt Publishing. 2022. 358 p.

5. Alyssa Miller. Cybersecurity Career Guide. Manning Publications. 2022. 368 p.

6. Awais Rashid, Howard Chivers, George Danezis, Emil Lupu, Andrew Martin. CyBOK The Cyber Security Body of Knowledge. The National Cyber Security Centre. 2019. 854 p.

7. Loren Kohnfelder. Designing Secure Software. No Starch Press. 2022. 332 p.

8. Samir Kumar Rakshit. Ethical Hacker's Penetration Testing Guide. BPB Online. 2022. 509 p.

9. Corey J. Ball. Hacking APIs. No Starch Press. 2022. 353 p.

10. Kevin Beaver. Hacking for Dummies. John Wiley & Sons. 2022. 419 p.

11. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p

12. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.

13. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.

14. Alex Matrosov, Eugene Rodionov, Sergey Bratus. Rootkits and Bootkits. No Starch Press. 2019. 450 p.

15. Петрик В.М., Присяжнюк М.М., Аль-Файюмі Халед та ін. «Системи інформаційної зброї та технології інформаційної війни»: підручник /

Петрик В.М., Присяжнюк М.М., Аль-Файюмі Халед, Жарков Я.М., Смірнов О.А, Буравченко К.О., Давидюк А.В., Кононович В.Г., Корчинский В.В., Кудирко В.М., Фесенко А.О.; за заг. ред. В.М. Петрика, М.М. Присяжнюка.– К.: Видавничий центр “Кафедра”, 2025.– 320 с.

16. Усік, П.С., Смірнова, Т.В., Буравченко, К.О., Смірнов, О.А., Улічев, О.С., Смірнов, С.А. «Дослідження технологій забезпечення кібербезпеки банківських систем з використанням штучного інтелекту». *Кібербезпека: освіта, наука, техніка.* 2025. Том 1 № 29. С.704–716, 2025

17. Kuznetsov, O., Frontoni, E., Kuznetsova, K., Arnesano, M., Smirnov, O. «A secure biometric authentication architecture for blockchain-driven cyber-physical systems». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 193–224.

18. Kuznetsov, O., Atzeni, G., Arnesano, M., Randieri, C., Smirnov, O. «Secure IoT-based smart wheelchair system: From implementation to security enhancement strategy». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 225–257.

19. Kuznetsov, O., Smirnov, O., Kuznetsova, T., Shaikhanova, A., Svatowsky, I. «Privacy-utility trade-offs in IoT networks: A comparative analysis of differential privacy mechanisms for sensor data aggregation». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 589–622.

20. 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.

21. Kuznetsov O., Frontoni E., Kuznetsova Y., Smirnov O., Moskovchenko I. «Trust-Based Security Architecture for Edge Computing: A Simulation Study of Dynamic Trust Evolution and Attack Detection». *CEUR Workshop Proceedings,* 2024,

3909, pp. 227–241.

22. 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.

23. 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.

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. Akhalaia, G., Iavich, M., Iashvili, G., Prysiazhnyy, D., Smirnova, T. «Secure Encrypted Connection on Georgian Website». *CEUR Workshop*

Proceedings, 2023, 3550, pp. 313-320.

30. 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

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. Козлов Я.О., Смірнова Т.В., Смірнов О.А. «Дослідження SIEM-систем для забезпечення кібербезпеки». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя*

кафедри кібербезпеки та програмного забезпечення, м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 26.

37. Козлов Я.О., Козірова Н.Л., Смірнов О.А. «Дослідження структури та принципу роботи SIEM-системи». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 59.

38. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 2(72), С. 170-178.

39. 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,

40. 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.

41. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

42. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

43. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І.,

Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

44. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

45. 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

46. 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.

47. 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.

48. 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.*

49. 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.

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

51. 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.

52. 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.

53. 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.