Міністерство освіти і науки України

Національний технічний університет

«Дніпровська політехніка»

Інститут електроенергетики

Факультет інформаційних технологій

Кафедра безпеки інформації та телекомунікацій

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи ступеня бакалавра

студенту *Явтушенко Артема Олексійовича*

академічної групи *125-18-2*

спеціальності *125 Кібербезпека*

спеціалізації

за освітньо - професійною програмою *Кібербезпека*

на тему *Метод протидії атаці "людина посередині" в корпоративній ІТС*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	доцент Герасіна О.В.	80	добре	
розділів:				
спеціальний	ас. Мілінчук Ю.А.	80	добре	
економічний	к.е.н., доц. Пілова Д.П.	70	задовільно	

Рецензент				

Нормоконтролер	ст. викладач Тимофєєв Д.С.			

Дніпро

2022

ЗАТВЕРДЖЕНО:
завідувач кафедри
безпеки інформації та телекомунікацій
_____ д.т.н., проф. Корнієнко В.І.

«_____»_____20___року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня бакалавра

студенту *Явтушенко Артема Олексійовича*
академічної групи *125-18-2*
спеціальності *125 Кібербезпека*
за освітньо - професійною програмою *Кібербезпека*
на тему *Метод протидії атаці "людина посередині" в корпоративній ІТС*
затверджену наказом ректора НТУ «Дніпровська політехніка» від _____ № _____

Розділ	Зміст	Термін виконання
Розділ 1	Огляд основних понять предметної області.	04.03.2022 – 25.03.2022
Розділ 2	Проектування і розробка додатку для детекції атак типу «людина посередині» на корпоративні мережі	22.04.2022 – 15.05.2022
Розділ 3	Економічний аналіз доцільності розробки системи	15.05.2022 – 16.06.2022

Завдання видано _____ _____
(підпис керівника) (прізвище, ініціали)

Дата видачі: _____

Дата подання до екзаменаційної комісії: _____

Прийнято до виконання _____ Явтушенко А.О
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 62с., 5 рис., 7 табл., 25 джерел.

Об'єкт розробки — корпоративні ІТС.

Предмет розробки — методи протидії атакам в корпоративних ІТС.

Метою роботи є розробка методу протидії атаці типу «людина посередині» на корпоративні ІТС.

У першому розділі розглянуте поняття корпоративної мережі та кібератаки. Детально роглянуті два основних види корпоративних мереж: "Зірка" та "Змішана", а також види кібератак.

Проведено аналіз існуючих загроз та методів боротьби з ними для корпоративної мережі типу "Зірка" та сформовано постановку задачі.

У другому розділі проведено проектування і розробку додатку детекції атак типу «людина посередині» на корпоративні мережі та експериментальне дослідження роботи додатку з інсцінюванням атаки порушника на корпоративну ІТС для підтвердження роботи алгоритму. Наведено інструкції для проведення атаки порушником та інструкції для користування додатком для виявлення вторгнень.

Третій розділ присвячено економічному аналізу доцільності розробки системи детекції атак типу «людина посередині» на корпоративні мережі.

Практична цінність та конкурентоздатність даного методу протидії атаці "людина посередині" полягає в перевазі в порівнянні з аналогами, зокрема: у надійності, простоті використання та гнучкості та вигідності розробки: отримано додатній економічний ефект у розмірі 56738,1грн.

ІНФОРМАЦІЙНА БЕЗПЕКА, НЕСАНЦІОНОВАНИЙ ДОСТУП, ЛЮДИНА ПОСЕРЕДИНІ, КІБЕРАТАКИ, КОРПОРАТИВНІ МЕРЕЖІ, ВИЯВЛЕННЯ ВТОРГНЕНЬ.

ABSTRACT

Explanatory note: 62c., 5 fig., 7 tables., 25 sources. The object of development is corporate ITS. The subject of development - methods of countering attacks in corporate ITS. The aim of the work is to develop a method of counteracting the attack of the type "man in the middle" on corporate ITS. The first section discusses the concept of corporate network and cyber attacks. Two main types of corporate networks are considered in detail: "Star" and "Mixed", as well as types of cyber attacks. An analysis of existing threats and methods of combating them for a corporate network such as "Star" and formed a problem statement. The second section designed and developed an application for detecting "man in the middle" attacks on corporate networks and an experimental study of the application with the instinct of an attacker's attack on corporate ITS to confirm the algorithm. Instructions for intruder attack and instructions for using the intrusion detection application are provided. The third section is devoted to the economic analysis of the feasibility of developing a system for detecting attacks such as "man in the middle" on corporate networks. The practical value of the competitiveness of this method of counteracting the attack "man in the middle" is the advantage over analogues, in particular: reliability, ease of use, flexibility and profitability of development: a positive economic effect of 56738.1 UAH. INFORMATION SECURITY, UNAUTHORIZED ACCESS, MAN IN THE MIDDLE, CYBER ATTACKS, CORPORATE NETWORKS, DETECTION OF INVASIONS.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

IDS — Intrusion Detection System

NIST — National Institute of Standards and Technology

OWSAP — Open Web Application Security Project

ЗМІСТ

ВСТУП

В зв’язку зі швидким розвитком інформаційних технологій і стрімким стрибком технологічного прогресу вперед такі речі як комп’ютер та інтернет давно перестали бути чимось, що виходить за рамки розуміння і можливостей звичайного обивателя, та стали чимось звичайним, зрозумілим і звичним.

Але, нажаль, зі стрибком популярності і доступності різного роду гаджетів і технологій — людина почала зберігати в електронному вигляді найрізноманітніші файли, які можуть містити дуже конфіденційну інформацію, потрапляння якої в чужі руки може призвести до величезних проблем.

Виходячи з усього вищесказаного, можна зробити висновок про необхідність спеціальних систем, які будуть запобігати крадіжкам особистих даних. Такі системи поділяються на підрозділи, в залежності від методу крадіжки, від якого система має вберегти. В даному конкретному випадку будуть розглядатися системи, які покликані вберегти користувача від вторгнення в систему за допомогою різного роду софту, який здатен надавати віддаленому персональному комп’ютеру доступ до системи жертви.

Виходячи з усього вищесказаного, зважаючи на описані наслідки крадіжки даних, важко недооцінити надвисоку актуальність подібного роду систем у сучасному світі.

Мета і завдання роботи обиралися повністю спираючись на все вище зазначене. Метою роботи є розробка програмного забезпечення для виявлення атак типу «людина посередині» на корпоративні мережі. Для виконання поставленої мети слід дослідити способи вторгнення в систему і маркери, які можуть дати розуміння про те, що відбувається вторгнення, або може відбутися в найближчий час.

Об’єкт дослідження — корпоративні мережі.

Предмет дослідження — атаки на корпоративні мережі.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття корпоративної мережі

Корпоративна мережа — це мережа, головним призначенням якої є підтримка роботи конкретного підприємства, що володіє даною мережею. Користувачами корпоративної мережі є тільки співробітники даного підприємства. На відміну від мереж операторів зв'язку, корпоративні мережі, в загальному випадку, не надають послуг стороннім організаціям або користувачам. Залежно від масштабу підприємства, а також від складності і різноманіття вирішуваних завдань розрізняють мережі відділу, мережі кампусу і корпоративні мережі[1] (термін «корпоративні» в даній класифікації набуває вузького значення — мережу великого підприємства).

Концепція корпоративної мережі

Будь-яка організація - це сукупність взаємодіючих елементів (підрозділів), кожен з яких може мати свою структуру. Елементи зв'язані між собою функціонально, тобто вони виконують окремі види робіт в рамках єдиного бізнес процесу, а також інформаційно, обмінюючись документами, факсами, письмовими і усними розпорядженнями і так далі крім того, ці елементи взаємодіють із зовнішніми системами, причому їх взаємодія також може бути як інформаційною, так і функціональною. І ця ситуація справедлива практично для всіх організацій, яким би видом діяльності вони не займалися - для урядової установи, банку, промислового підприємства, комерційної фірми і так далі.

Такий загальний погляд на організацію дозволяє сформулювати деякі загальні принципи побудови корпоративних інформаційних систем, тобто інформаційних систем в масштабі всієї організації.

Призначення корпоративної мережі

Корпоративною мережею вважається будь-яка мережа, що працює по протоколу TCP/IP[2]. і використовує комунікаційні стандарти Інтернету, а також сервісні застосування, що забезпечують доставку даних користувачам мережі. Наприклад, підприємство може створити сервер Web для публікації оголошень, виробничих графіків і інших службових документів. Службовці здійснюють доступ до необхідних документів за допомогою засобів (коштів) проглядання Web.

Сервери Web корпоративної мережі можуть забезпечити користувачам послуги, аналогічні послугам Інтернету, наприклад роботу з гіпертекстовими сторінками (що містять текст, гіперпосилання, графічні зображення і звукозаписи), надання необхідних ресурсів по запитах клієнтів Web, а також здійснення доступу до баз даних. У цьому керівництві всі служби публікації називаються “Службами Інтернету” незалежно від того, де вони використовуються (у Інтернеті або корпоративній мережі).

Корпоративна мережа, як правило, є територіально розподіленою, тобто об'єднуючою офіси, підрозділи і інші структури, що знаходяться на значному віддаленні один від одного. Принципи, по яких будується корпоративна мережа, досить сильно відрізняються від тих, що використовуються при створенні локальної мережі. Це обмеження є принциповим, і при проектуванні[3].

Процес створення корпоративної інформаційної системи

Можна виділити основні етапи процесу створення корпоративної інформаційної системи:

- провести інформаційне обстеження організації;
- за результатами обстеження вибрати архітектуру системи і апаратно-програмні засоби її реалізації;
- за результатами обстеження вибрати і/або розробити ключові компоненти інформаційної системи;

Інформаційне обстеження

Інформаційна система потрібна організації для того, щоб забезпечувати інформаційно-комунікаційну підтримку її основної і допоміжної діяльності. Тому перш, ніж вести мову про структуру і функціональне наповнення інформаційної системи, необхідно розібратися в цілях і завданнях самої організації, аби зрозуміти, що ж потрібно автоматизувати.

Цілі інформаційного обстеження:

- формулювання і опис функцій кожного підрозділу компанії, а також вирішувані ними завдання;
- опис технології роботи кожного з підрозділів компанії і розуміння, що необхідно автоматизувати і в якій послідовності;
- опис технології роботи кожного з підрозділів і пов'язаних з ними інформаційних потоків;
- відображення технології на структуру, визначення її функціонального складу і кількості робочих місць в кожному структурному підрозділі компанії, а також опис функцій, які виконуються (автоматизуються) на кожному робочому місці;
- опис основних шляхів і алгоритми проходження вхідних, внутрішніх і витікаючих документів, а також технології їх обробки.

Результатом обстеження є моделі діяльності компанії, і її інформаційної інфраструктури, на базі яких розробляються проект корпоративної інформаційної системи, вимоги до програмно-апаратних засобів і специфікації на розробку прикладного програмного забезпечення, якщо в цьому є необхідність.

Вибір архітектури

Для корпоративних систем рекомендується архітектура клієнт/сервер. Архітектура клієнт/сервер надає технологію доступу кінцевого користувача до інформації в масштабах підприємства. Таким чином, архітектура клієнт/сервер дозволяє створити єдиний інформаційний простір, в якому

кінцевий користувач має своєчасний і безперешкодний (але санкціонований) доступ до корпоративної інформації.

Вибір СУБД

Вибір системи управління для корпоративної бази даних - один з ключових моментів в розробці інформаційної системи. На Російському ринку присутні практично всі СУБД, що належать до елітного класу - Oracle, Informix, Sybase, Ingres. Питання, яку СУБД використовувати, можна вирішити тільки за результатами попереднього обстеження і отримання інформаційних моделей діяльності.

Структура корпоративної мережі

Сервера об'єднуються в мережу, що дозволяє передавати інформацію в корпоративному середовищі від одного користувача до іншого. Корпоративна мережа може включати одну і більше організацій. Одну організацію може обслуговувати один сервер, кілька серверів або один сервер може обслуговувати кілька організацій. Можливі змішані схеми, що включають всі вищеперелічені випадки.

Мережа та система адресації в мережі підтримує рекомендації протокола X-400. Даний протокол використовується у випадку вимог підвищеної надійності до передачі даних.

Для передачі повідомлень між серверами і автоматизованим робочим місцем використовується протокол TcpFoss, між сервером і інтерактивним клієнтом протокол CORBA, між сервером і Web-клієнтом протокол HTTPS.

1.2 Конфігурації корпоративних мереж

Топологія мережі “Зірка”

Для забезпечення керованості і надійності роботи мережі її необхідно будувати по топології “ЗІРКА” з урахуванням того, що кожен промінь “ЗІРКИ” повинен бути, в свою чергу, теж зіркою. Така топологія дозволяє управляти розташованими нижче по ієрархії серверами сервером, який розташований вище, що підвищує надійність і керованість мережі. Також, це гарантує наявність “третьої сторони” при передачі повідомлень від користувача, що обслуговується одним сервером до користувача, який обслуговується іншим сервером. Це важливо, наприклад, при побудові банківських мереж чи мереж з підвищеною відповідальністю користувачів за своєчасну передачу інформації.

Наприклад, поштове повідомлення від клієнта сервера, який обслуговує Організацію 3, клієнту сервера, який обслуговує Організацію 2, проходить через сервер Організації 1.

Побудова мережі з багатьма вузлами з одного боку збільшує експлуатаційні витрати і ускладнює її обслуговування, але з іншого боку, знижує навантаження на сервера мережі і канали, а також спрощує роботу адміністратора. Не потрібно адмініструвати величезну кількість користувачів на одному сервері, тим більше співробітників іншої Організації.

Здається доцільним використати кілька серверів FossDocMail для обслуговування однієї організації у випадках, якщо її підрозділи територіально рознесені і мають велику кількість персоналу, а для передачі інформації використовуються мережі загального доступу. Така топологія підвищує захищеність і надійність мережі.

Топологія мережі “Змішана”

Сервера, що знаходяться на одному рівні, можуть передавати повідомлення безпосередньо один одному. При такій організації мережі знижується навантаження на канали передачі даних, але керованість мережі

різко падає. Також відсутня “третя сторона“, яка може виступати арбітром при виникненні спірних ситуацій між різними Організаціями і підрозділами Організації.

Можливі й інші варіанти топології мережі, але всі вони будуть похідними від описаних вище. Топологія мережі залежить від налаштування маршрутних таблиць на вузлах мережі.

1.3 Кібератаки і їх види

Кібератака — це будь-який наступальний маневр, спрямований на комп'ютерні інформаційні системи, комп'ютерні мережі, інфраструктуру чи пристрої персональних комп'ютерів.[1] Зловмисник – це особа або процес, який намагається отримати доступ до даних, функцій або інших обмежених областей системи без авторизації, потенційно зі зловмисним наміром.[2] Залежно від контексту, кібератаки можуть бути частиною кібервійни або кібертероризму. Кібератака може бути використана суверенними державами, окремими особами, групами, суспільством або організаціями, і вона може походити з анонімного джерела. Продукт, який сприяє кібератаці, іноді називають кіберзброєю.

Кібератака може вкрасти, змінити або знищити певну ціль шляхом злому в сприйнятливу систему.[3] Кібератаки можуть варіюватися від встановлення шпигунських програм на персональний комп'ютер до спроб знищити інфраструктуру цілих країн. Юридичні експерти намагаються обмежити використання цього терміну інцидентами, які завдають фізичної шкоди, відрізняючи його від більш звичайних зломів даних і більш широких хакерських дій.[4]

Кібератаки стають дедалі складнішими та небезпечними.[5]

Щоб запобігти цим атакам, можна використовувати аналітику поведінки користувачів і управління інформацією та подіями безпеки (SIEM).

Кібервійна використовує методи захисту та атаки на інформаційні та комп'ютерні мережі, які населяють кіберпростір, часто через тривалу кіберкампанію або серію пов'язаних кампаній. Він заперечує здатність супротивника робити те ж саме, використовуючи технологічні інструменти війни для атаки на критичні комп'ютерні системи супротивника. З іншого боку, кібертероризм — це «використання інструментів комп'ютерної мережі для закриття критично важливих національних інфраструктур (таких як енергетика, транспорт, урядові операції) або для примусу чи залякування

уряду чи цивільного населення».[16] Це означає, що як кібервійна, так і кібертероризм є одним і тим же результатом, щоб пошкодити критичні інфраструктури та комп'ютерні системи, пов'язані між собою в межах кіберпростору.

Експерт з фінансових злочинів Вейт Бюттерлін пояснив, що організації, у тому числі державні суб'єкти, які не можуть фінансувати себе за рахунок торгівлі через введені санкції, здійснюють кібератаки на банки для отримання коштів.[17]

Атака може бути активною або пасивною.[7]

«Активна атака» намагається змінити системні ресурси або вплинути на їх роботу.

«Пасивна атака» намагається дізнатися або використати інформацію з системи, але не впливає на системні ресурси (наприклад, прослуховування телефонних розмов).

Атака може бути здійснена як інсайдером, так і не в організації;[7]

«Внутрішня атака» — це атака, ініційована сутністю всередині периметра безпеки («інсайдер»), тобто об'єктом, який має дозвіл на доступ до системних ресурсів, але використовує їх у спосіб, не схвалений тими, хто надав дозвіл.

«Зовнішня атака» ініціюється з-за периметра неавторизованим або незаконним користувачем системи («аутсайдер»). В Інтернеті потенційні сторонні нападники варіюються від шанувальників-любителів до організованих злочинців, міжнародних терористів і ворожих урядів.[7]

Ресурс (як фізичний, так і логічний), який називається активом, може мати одну або кілька вразливостей, які можуть бути використані агентом загрози під час дії загрози. В результаті може бути порушена конфіденційність, цілісність або доступність ресурсів. Потенційно, збиток може поширюватися на ресурси на додаток до тих, які спочатку були

визначені як уразливі, включаючи додаткові ресурси організації та ресурси інших залучених сторін (замовників, постачальників).

Основою інформаційної безпеки є так звана тріада ЦРУ.

Атака може бути активною, коли вона намагається змінити системні ресурси або вплинути на їх роботу: таким чином вона ставить під загрозу цілісність або доступність. «Пасивна атака» намагається дізнатися або використати інформацію з системи, але не впливає на системні ресурси: таким чином, вона ставить під загрозу конфіденційність.

Загроза – це потенційна можливість порушення безпеки, яка існує, коли є обставини, можливості, дія чи подія, які можуть порушити безпеку та завдати шкоди. Тобто загроза — це можлива небезпека, яка може використати вразливість. Загроза може бути або «навмисною» (тобто розумною; наприклад, окремий зломщик або злочинна організація), або «випадковою» (наприклад, можливість несправності комп'ютера або можливість «діяння Бога», наприклад землетрус, пожежа чи смерч).[7]

Набір політик, що стосуються управління інформаційною безпекою, системи управління інформаційною безпекою (СУІБ), був розроблений для управління, відповідно до принципів управління ризиками, контрзаходами з метою досягнення стратегії безпеки, створеної відповідно до правил і положень, що застосовуються в країна.[22]

Атака повинна призвести до інциденту безпеки, тобто події безпеки, яка передбачає порушення безпеки. Іншими словами, системна подія, що має відношення до безпеки, під час якої політика безпеки системи порушується або іншим чином порушується.

Загальна картина представляє фактори ризику сценарію ризику.[23]

Організація повинна вжити заходів для виявлення, класифікації та управління інцидентами безпеки. Першим логічним кроком є створення плану реагування на інцидент і, зрештою, комп'ютерної групи реагування на надзвичайні ситуації.

Для виявлення атак на організаційному, процедурному та технічному рівнях можна запровадити ряд контрзаходів. Прикладами є група реагування на надзвичайні ситуації з комп'ютером, аудит безпеки інформаційних технологій та система виявлення вторгнень.[24]

Атаку зазвичай здійснює хтось із поганими намірами: до цієї категорії належать атаки з чорним капелюхом, тоді як інші проводять тестування на проникнення в інформаційній системі організації, щоб з'ясувати, чи діють усі передбачені засоби контролю.

Атаки можна класифікувати за походженням, тобто якщо вони проводяться за допомогою одного або кількох комп'ютерів: в останньому випадку це називається розподіленою атакою. Ботнети використовуються для проведення розподілених атак.

Інші класифікації залежать від використовуваних процедур або типу використовуваних уразливостей: атаки можуть бути зосереджені на мережевих механізмах або функціях хоста.

Деякі атаки є фізичними: тобто крадіжка або пошкодження комп'ютерів та іншого обладнання. Інші — це спроби примусово змінити логіку, яку використовують комп'ютери або мережеві протоколи, щоб досягти непередбачуваного (первісним дизайнером) результату, але корисного для зловмисника. Програмне забезпечення, яке використовується для логічних атак на комп'ютери, називається шкідливим ПЗ.

1.4 Атака «людина посередині»

У криптографії та комп'ютерній безпеці людина посередині, монстр-по-середині, [1][2] машина-по-середині, мавпа-по-середині [3]. Атака «посередині» [4] (MITM) або «person-in-the-middle» [5] (PITM) — це кібератака, коли зловмисник таємно передає і, можливо, змінює комунікації між двома сторонами, які вважають, що вони безпосередньо спілкуються з один одного, оскільки нападник встався між двома сторонами.[6] Одним із прикладів атаки MITM є активне підслуховування, під час якого зловмисник встановлює незалежні зв'язки з жертвами та передає повідомлення між ними, щоб змусити їх повірити, що вони розмовляють безпосередньо один з одним через приватне з'єднання, хоча насправді вся розмова контролюється нападника.[7] Зловмисник повинен мати можливість перехопити всі відповідні повідомлення, що передаються між двома жертвами, і ввести нові. Це просто за багатьох обставин; наприклад, зловмисник в межах діапазону прийому незашифрованої точки доступу Wi-Fi може вставити себе як людина посередині.[8][9][10] Оскільки атака MITM має на меті обійти взаємну аутентифікацію, вона може бути успішною лише тоді, коли зловмисник достатньо добре імітує кожну кінцеву точку, щоб задовольнити їхні очікування. Більшість криптографічних протоколів включають певну форму аутентифікації кінцевої точки спеціально для запобігання атакам MITM. Наприклад, TLS може автентифікувати одну або обидві сторони за допомогою взаємно довіреного центру сертифікації.[11][9]

1.5 Існуючі варіанти вирішення проблеми

Атаки MITM можна запобігти або виявити двома способами: аутентифікацією та виявленням несанкціонованого доступу. Аутентифікація забезпечує певний ступінь впевненості, що дане повідомлення надійшло з законного джерела. Виявлення несанкціонованого доступу лише показує докази того, що повідомлення, можливо, було змінено.

Аутентифікація

Усі криптографічні системи, захищені від атак MITM, забезпечують певний метод аутентифікації для повідомлень. Більшість вимагає обміну інформацією (наприклад, відкритими ключами) на додаток до повідомлення через захищений канал. Такі протоколи, які часто використовують протоколи узгодження ключа, були розроблені з різними вимогами безпеки для захищеного каналу, хоча деякі намагалися скасувати вимоги щодо будь-якого безпечного каналу взагалі.[13]

Інфраструктура відкритих ключів, така як захист транспортного рівня, може посилити протокол керування передачею проти атак MITM. У таких структурах клієнти та сервери обмінюються сертифікатами, які видаються та перевіряються довіреною третьою стороною, яка називається центром сертифікації (ЦС). Якщо оригінальний ключ для аутентифікації цього ЦС сам по собі не був об'єктом атаки MITM, тоді сертифікати, видані ЦС, можуть використовуватися для аутентифікації повідомлень, надісланих власником цього сертифіката. Використання взаємної аутентифікації, за якої і сервер, і клієнт перевіряють комунікацію іншого, охоплює обидві сторони атаки MITM. Якщо ідентичність сервера або клієнта не підтверджена або визнана недійсною, сеанс завершиться.[14] Однак поведінка більшості з'єднань за замовчуванням полягає лише в автентифікації сервера, що означає, що взаємна автентифікація не завжди використовується, і атаки MITM все ще можуть відбуватися.

Атестації, такі як вербальні комунікації спільного значення (як у ZRTP), або записані атестації, такі як аудіо/візуальні записи хешу відкритого ключа[15], використовуються для запобігання атакам MITM, оскільки візуальні медіа є набагато складнішими та займають багато часу. - наслідувати набагато важче, ніж просте передача пакетів даних. Однак ці методи вимагають, щоб людина була в циклі, щоб успішно ініціювати транзакцію.

У корпоративному середовищі успішна автентифікація (як вказує зелений замок браузера) не завжди означає безпечне з'єднання з віддаленим сервером. Корпоративна політика безпеки може передбачати додавання спеціальних сертифікатів у веб-браузери робочих станцій, щоб мати можливість перевіряти зашифрований трафік. Як наслідок, зелений навісний замок не означає, що клієнт успішно пройшов автентифікацію на віддаленому сервері, а лише на корпоративному сервері/проксі, що використовується для перевірки SSL/TLS.

Закріплення відкритого ключа HTTP (HPKP), яке іноді називають «закріпленням сертифіката», допомагає запобігти атаці MITM, під час якої скомпрометовано сам центр сертифікації, оскільки сервер надає список «закріплених» хешів відкритих ключів під час першої транзакції. Наступні транзакції вимагають, щоб один або кілька ключів у списку були використані сервером для автентифікації цієї транзакції.

DNSSEC розширює протокол DNS для використання підписів для аутентифікації записів DNS, запобігаючи простим атакам MITM, спрямованим клієнта на шкідливу IP-адресу.

Виявлення несанкціонованого доступу

Перевірка затримки потенційно може виявити атаку в певних ситуаціях [16], наприклад, при довгих обчисленнях, які тривають десятки секунд, як-от хеш-функції. Щоб виявити потенційні атаки, сторони перевіряють на наявність розбіжностей у часі відповіді. Наприклад: скажімо, що двом

сторонам зазвичай потрібно певний час для виконання певної транзакції. Однак, якщо для однієї транзакції знадобиться ненормальний час, щоб досягти іншої сторони, це може свідчити про втручання третьої сторони, що вносить додаткову затримку в транзакцію.

Теоретично квантова криптографія забезпечує докази фальсифікації транзакцій за допомогою теореми про відсутність клонування. Протоколи, засновані на квантовій криптографії, зазвичай аутентифікують частину або всю їх класичну комунікацію за допомогою безумовно безпечної схеми аутентифікації. Як приклад аутентифікації Вегмана-Картера.[17]

1.6 Постановка задачі

Обрана корпоративна мережа має конфігурацію «зірка» та керується ОС “Windows”, в якій кожний окремий «промінь» — це канал зв’язку з іншим, підлеглим сервером.

Топологічна схема мережі зображена на рисунку 1.1

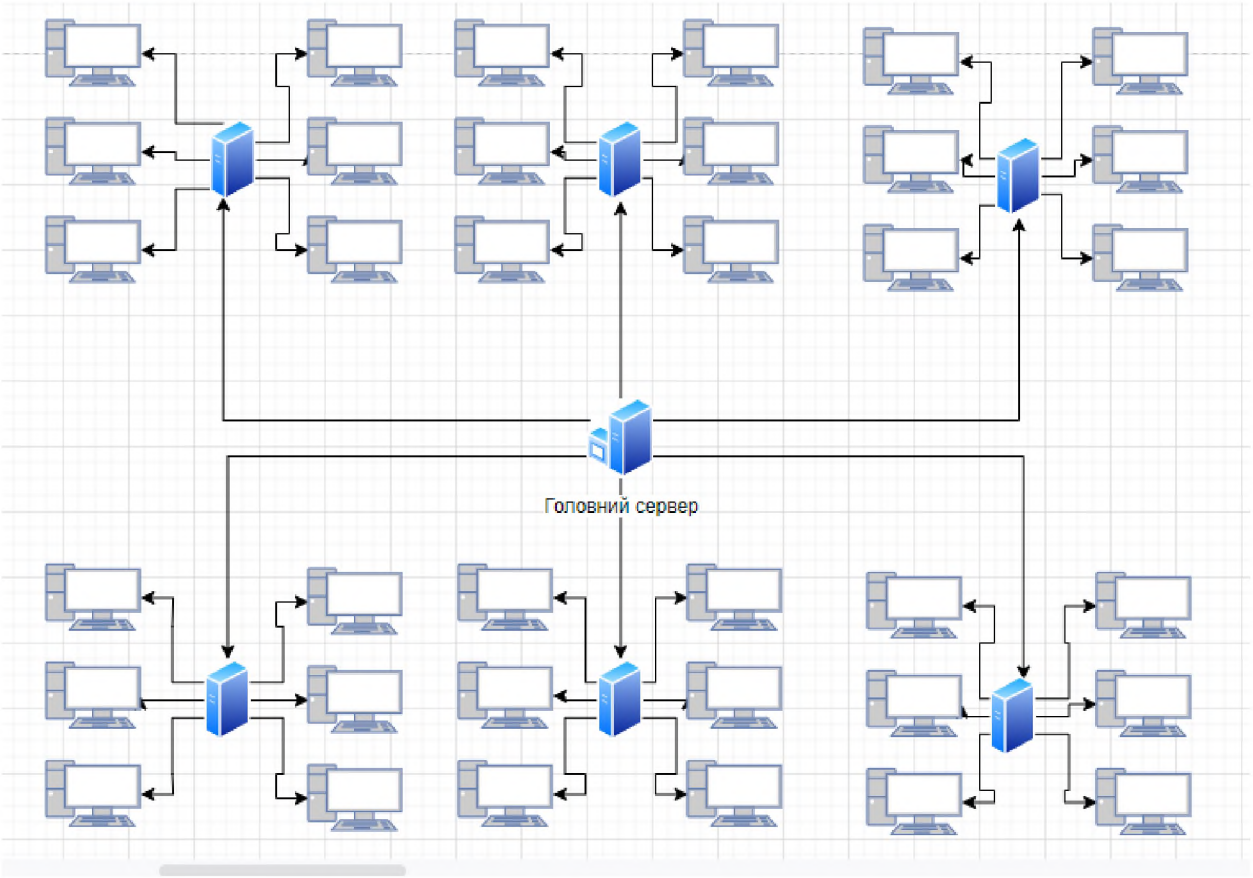


Рисунок 1.1 — Схема корпоративної мережі

Оскільки даний канал може мати проріхи в системі безпеки — кожен з таких каналів може стати ціллю атаки «людина посередині» при якій зловмисник зможе перехоплювати всі дані, що передаються від основного сервера до периферичних. Таким чином, основна задача полягає в розробленні програмного модуля виявлення атаки «людина посередині» для ОС “Windows”, який буде використовуватися на головному сервері мережі та у інсцінуванні атаки на корпоративну мережу для перевірки його працездатності.

Висновки до розділу 1

В ході написання першого розділу було проаналізовано основні поняття предметної області, такі як корпоративні мережі та кібератаки, виявлено основні види корпоративних мереж і кібератак, детально розглянуто поняття атаки «людина посередині» і механізм її роботи.

Проведено огляд існуючих засобів виявлення і боротьби з атаками типу «людина посередині».

Результатом розділу стала постановка задачі на розробку програмного забезпечення для виявлення атак типу «людина посередині» з метою підвищення рівня безпеки корпоративної мережі з топологією «зірка».

РОЗДІЛ 2
ПРОЕКТУВАННЯ І ПРОГРАМНА РЕАЛІЗАЦІЯ

2.1 Огляд інструментальних засобів розробки

C # (вимовляється як музична нота C♯, але написаний зі знаком числа) - це універсальна багатопарадігмальна мова програмування, що включає строгу типізацію, лексичну область видимості, імператив, декларативний, функціональний, універсальний об'єкт -орієнтовані (на основі класів) та компонентно-орієнтовані дисципліни програмування. Він був розроблений Microsoft приблизно в 2000 році в рамках ініціативи .NET, а потім затверджений в якості міжнародного стандарту Ecma (ECMA-334) і ISO (ISO / IEC 23270: 2018). Mono - це назва безкоштовного проекту з відкритим вихідним кодом для розробки компілятора і середовища виконання для мови. C # є одним з мов програмування, розроблених для Common Language Infrastructure (CLI).

З часу випуску C # 2.0 в листопаді 2005 року мови C # і Java розвивалися за все більш і більш розбіжним траєкторіях, ставши двома зовсім різними мовами. Одним з перших основних відхилень стало додавання узагальнень до обох мов з абсолютно різними реалізаціями. C # використовує reification для надання «першокласних» універсальних об'єктів, які можна використовувати як будь-який інший клас, з генерацією коду, що виконується під час завантаження класу. Крім того, в C #, додано кілька основних функцій для програмування функціонального стилю, кульмінацією яких є розширення LINQ, випущені в C # 3.0, і підтримуюча його структура лямбда-виразів, методів розширення і анонімних типів. Ці функції дозволяють програмістам C # використовувати методи функціонального програмування, такі як замикання, коли це вигідно для їх застосування. Розширення LINQ і функціональний імпорт допомагають розробникам

скоротити обсяг стандартного коду, який включається в загальні завдання, такі як запити до бази даних, аналіз файлу XML або пошук в структурі даних, зміщуючи акцент на реальну логіку програми, щоб поліпшити читаність і ремонтопридатність.

По своєму дизайну C # є мовою програмування, який безпосередньо відображає основну інфраструктуру спільної мови (CLI). Більшість його внутрішніх типів відповідають типам значень, реалізованих середовищем CLI. Однак специфікація мови не містить вимог до генерації коду компілятора, тобто не вказує, що компілятор C # має орієнтуватися на середовище виконання спільної мови, або генерувати загальний проміжний мова (CIL), або генерувати будь-який інший конкретний формат. Теоретично, компілятор C # може генерувати машинний код, як традиційні компілятори C ++ та Fortran.

C # підтримує строго типізовані оголошення неявних змінних з ключовим словом var, а також неявно типізовані масиви з ключовим словом new [], за яким слід ініціалізатор колекції.

C # підтримує суворий логічний тип даних, bool. Оператори, які приймають умови, такі як while і if, вимагають вирази типу, що реалізує оператор true, такого як логічний тип. Хоча C ++ також має логічний тип, він може вільно перетворюватися в цілі і з цілих чисел, а вирази, наприклад, якщо (a) вимагають тільки, щоб a був перетворений в bool, дозволяючи a бути int або покажчиком. C # забороняє цей підхід «цілочисельне значення істина або брехня» на тій підставі, що примус програмістів використовувати вирази, які повертають саме bool, може запобігти певні типи помилок програмування, таких як if (a = b) (використання присвоювання = замість рівності ==).

C # більш безпечний для типів, ніж C ++. Єдиними неявними перетвореннями за замовчуванням є ті, які вважаються безпечними, наприклад, розширення цілих чисел. Це застосовується під час компіляції,

під час JIT і, в деяких випадках, під час виконання. Не відбувається неявного перетворення між логічними значеннями і цілими числами, а також між членами перерахування і цілими числами (за винятком літерала 0, який може бути неявно перетворений в будь-який перераховується тип). Будь кероване перетворення повинне бути чітко позначено як явне або неявне, на відміну від конструкторів копіювання C ++ та операторів перетворення, які за замовчуванням неявні.

C # має явну підтримку коваріації і контраваріантності в універсальних типах, на відміну від C ++, який має деяку ступінь підтримки контраваріантності просто завдяки семантиці повертаються типів віртуальних методи.

Учасники перерахування розміщуються у своїй області видимості.

Мова C # не допускає глобальних змінних або функцій. Всі методи і члени повинні бути оголошені всередині класів. Статичні члени відкритих класів можуть замінювати глобальні змінні і функції.

Локальні змінні не можуть приховувати змінні вміщує блоку, на відміну від C та C ++.

C # має підтримку строго типізованих покажчиків на функції через ключове слово делегат. Як і псевдо-C ++ фреймворк Qt, в C # є семантика, зокрема навколишнє події стилю публікації-підписки, хоча C # використовує для цього делегати.

Керована пам'ять не може бути звільнена; замість цього він автоматично збирається. Збірка сміття вирішує проблему витоків пам'яті, позбавляючи програміста від відповідальності за звільнення пам'яті, яка більше не потрібна.

На відміну від C ++, C # не підтримує множинне успадкування, хоча клас може реалізовувати будь-яку кількість інтерфейсів. Це було дизайнерське рішення провідного архітектора мови, щоб уникнути ускладнення і спростити архітектурні вимоги у всьому CLI. При реалізації

декількох інтерфейсів, які містять метод з однаковою сигнатурою, і. тобто два методу з одним і тим же ім'ям, що приймають параметри одного й того ж типу в одному і тому ж порядку, C # дозволяє реалізовувати кожен метод в залежності від того, через який інтерфейс викликається цей метод, або, як Java, дозволяє реалізувати метод один раз і мати один виклик за викликом через будь-який з інтерфейсів класу.

Однак, на відміну від Java, C # підтримує перевантаження операторів. Тільки найбільш часто перевантажені оператори в C ++ можуть бути перевантажені в C #.

C # має можливість використовувати LINQ через .NET Framework. Розробник може запросити будь-який об'єкт IEnumerable <T>, документи XML, набір даних ADO.NET і бази даних SQL. Використання LINQ в C # дає такі переваги, як підтримка Intellisense, потужні можливості фільтрації, безпека типів з можливістю перевірки помилок компіляції і узгодженість даних для запиту з різних джерел. Є кілька різних мовних структур, які можна використовувати з C # з LINQ, і вони є виразами запитів, лямбда-виразами, анонімними типами, неявно типізованими змінними, методами розширення і ініціалізаторами об'єктів

Визначення мови C # і CLI стандартизовані у відповідності зі стандартами ISO і Ecma, які забезпечують розумну та недискримінаційну ліцензійну захист від патентних претензій.

Microsoft погодилася не пред'являти позов розробникам програмного забезпечення з відкритим вихідним кодом за порушення патентів в некомерційних проектах в частині структури, охоплюваній OSP. Microsoft також погодилася не застосовувати патенти на продукти Novell щодо платять клієнтів Novell, за винятком списку продуктів, в яких явно не згадується C#.NET або реалізація Novell .NET (The Mono Project). Тим не менш, Novell стверджує, що Mono не порушує жодних патентів Microsoft. Microsoft також уклала конкретну угоду про відмову у захисту патентних прав, пов'язаних з

плагіном браузера Moonlight, який залежить від Mono, якщо він отриманий через Novell

Microsoft очолює розробку еталонного компілятор C # з відкритим вихідним кодом і набору інструментів, що раніше носили кодова назва "Roslyn". Компілятор, який повністю написаний на керованому коді (C #), був відкритий, а функціональність відображена як API. Це дозволяє розробникам створювати інструменти рефакторінгу та діагностики.

Інші компілятори C # (деякі з яких включають реалізацію інфраструктури спільної мови і бібліотек класів .NET):

• Проект Mono надає компілятор C # з відкритим вихідним кодом, повну реалізацію загальномовної інфраструктури з відкритим вихідним кодом, включаючи необхідні бібліотеки інфраструктури, як вони зазначені в специфікації ECMA, і майже повну реалізацію пропрієтарних бібліотек класів Microsoft .NET до .NET 3.5. Починаючи з Mono 2.6, планів по впровадженню WPF не існує; WF планується до більш пізнього випуску; і є лише часткові реалізації LINQ to SQL і WCF.

• Проект DotGNU (в даний час припинений) також надав компілятор C # з відкритим вихідним кодом, майже повну реалізацію інфраструктури спільної мови, включаючи необхідні бібліотеки інфраструктури, як вони зазначені в специфікації ECMA, і підмножина деяких залишилися пропрієтарних класів Microsoft .NET. бібліотеки .NET 2.0 (не документовані або не включені в специфікацію ECMA, але включені в стандартний дистрибутив Microsoft .NET Framework).• Загальна мовна інфраструктура загального ресурсу Microsoft під кодовою назвою «Rotor» забезпечує реалізацію спільного джерела середовища CLR і компілятора C #, ліцензованого тільки для освітніх і дослідницьких цілей, а також підмножина необхідних бібліотек інфраструктури Common Language Infrastructure в специфікації ECMA (вгору в C # 2.0 і підтримується тільки в Windows XP).

C# була обрана основною мовою розробки даного проекту з огляду на усе вищеописане, а саме — на функціонал, який підтримує дана мова, операційні системи, які підтримуються та простота вивчення.

Загалом — функціонал C# найкраще підходить під дану розробку, за рахунок простої реалізації основних принципів ООП, простоти розробки форм та зрозумілого інтерфейсу.

Microsoft Visual Studio - це інтегроване середовище розробки (IDE) від Microsoft. Він використовується для розробки комп'ютерних програм, а також веб-сайтів, веб-додатків, веб-служб та мобільних додатків. Visual Studio використовує платформи Microsoft для розробки програмного забезпечення, такі як API Windows, Windows Forms, Foundation Presentation Foundation, Windows Store та Microsoft Silverlight. Він може створювати як власний код, так і керований код.

Visual Studio включає редактор коду, що підтримує IntelliSense (компонент доповнення коду), а також рефакторинг коду. Інтегрований налагоджувач працює як налагоджувач на рівні джерела, так і налагоджувач на рівні машини. Інші вбудовані інструменти включають кодовий профілер, конструктор для побудови програм GUI, веб-дизайнер, дизайнер класів та дизайнер схем бази даних. Він приймає плагіни, які покращують функціональність майже на кожному рівні - включаючи додавання підтримки для систем керування джерелами (наприклад, Subversion і Git) та додавання нових наборів інструментів, таких як редактори та візуальні дизайнери для мов, що задаються доменом або набори інструментів для інших аспектів розробки програмного забезпечення життєвий цикл (як клієнт Azure DevOps: Team Explorer).

Visual Studio підтримує 36 різних мов програмування та дозволяє редактору коду та налагоджувачу підтримувати (в різній мірі) майже будь-яку мову програмування за умови існування послуги, що залежить від мови. Вбудовані мови включають C, C ++, C ++ / CLI, Visual Basic .NET, C #, F #,

JavaScript, TypeScript, XML, XSLT, HTML та CSS. Підтримка інших мов, таких як Python, Ruby, Node.js та M серед інших, доступна через плагіни. Java (і J #) підтримувалися в минулому.

Найбільш основне видання Visual Studio, спільноти, доступне безкоштовно. Гасло для видання Visual Studio Community - "Безкоштовне повнофункціональне IDE для студентів, відкритих джерел та індивідуальних розробників".

На даний момент підтримується версія Visual Studio - 2019 рік.

Visual Studio не підтримує жодної мови програмування, рішення чи інструменту внутрішньо; натомість це дозволяє підключати функціональність, кодовану як VSPackage. Після встановлення функціональність доступна як Сервіс. IDE надає три послуги: SVsSolution, яка надає можливість перераховувати проекти та рішення; SVsUIShell, який забезпечує функцію вікон та інтерфейсу користувача (включаючи вкладки, панелі інструментів та вікна інструментів); та SVsShell, яка займається реєстрацією VSPackages. Крім того, IDE також відповідає за координацію та забезпечення зв'язку між службами. Всі редактори, дизайнери, типи проектів та інші інструменти реалізовані як VSPackages. Visual Studio використовує COM для доступу до VSPackages. SDK Visual Studio також включає в себе керовану рамку пакетів (MPF), яка є набором керованих обгорток навколо COM-інтерфейсів, які дозволяють писати пакети будь-якою мовою, сумісною з CLI. Однак MPF не забезпечує всю функціональність, що піддається інтерфейсам Visual Studio COM. Потім послуги можуть бути використані для створення інших пакетів, які додають функціональність Visual Studio IDE.

Підтримка мов програмування додається за допомогою спеціального VSPackage, який називається Language Service. Мовна служба визначає різні інтерфейси, які може реалізувати реалізація VSPackage, щоб додати підтримку різних функціональних можливостей. Функції, які можна додати таким чином, включають забарвлення синтаксису, завершення оператора,

відповідність дужок, підказки інформації про параметри, списки членів та маркери помилок для складання фону. Якщо інтерфейс буде реалізований, функціонал буде доступний для мови. Мовні послуги реалізуються на мовній основі. Реалізації можуть повторно використовувати код з аналізатора або компілятора для мови. Мовні сервіси можуть бути реалізовані як у рідному коді, так і керованому коді. Для нативного коду можуть бути використані або рідні інтерфейси COM, або Babel Framework (частина Visual Studio SDK). Для керованого коду MPF включає обгортки для написання сервісів керованої мови.

Visual Studio не включає вбудовану підтримку управління джерелами, але вона визначає два альтернативних способи інтеграції систем управління джерелами з IDE. VSPackage управління джерелом може забезпечити власний індивідуальний інтерфейс користувача. Навпаки, плагін управління джерелом за допомогою MSSCCI (Microsoft Source Code Control Interface) надає набір функцій, які використовуються для реалізації різних функцій управління джерелом, зі стандартним інтерфейсом користувача Visual Studio. MSSCCI вперше використовувався для інтеграції Visual SourceSafe з Visual Studio 6.0, але згодом був відкритий через SDK Visual Studio. Visual Studio .NET 2002 використовував MSSCCI 1.1, а Visual Studio .NET 2003 використовували MSSCCI 1.2. Visual Studio 2005, 2008 та 2010 використовує MSSCCI версії 1.3, яка додає підтримку перейменування та видалення розповсюдження, а також асинхронного відкриття.

Visual Studio підтримує запуск декількох екземплярів середовища (кожен зі своїм набором VSPackages). Екземпляри використовують різні вулики реєстру (див. Визначення MSDN терміна "вулик реєстру" у сенсі, що використовується тут) для зберігання конфігураційного стану та диференціюються їх AppId (ідентифікатор програми). Екземпляри запускаються специфічним для AppId .exe, який вибирає AppId, встановлює кореневий вулик та запускає IDE. VSP-пакети, зареєстровані для одного

AppId, інтегруються з іншими VSP-пакетами для цього AppId. Різні випуски продуктів Visual Studio створені за допомогою різних додатків. Продукти випуску Visual Studio Express встановлюються разом із власними AppIds, але продукти Standard, Professional та Team Suite мають однаковий AppId. Отже, видання Express можна встановити поряд з іншими виданнями, на відміну від інших видань, які оновлюють ту саму установку. Професійне видання включає в себе набір VSPackages у стандартному виданні, а командний набір включає суперсет VSPackages в обох інших виданнях. Система AppId використовується Visual Studio Shell в Visual Studio 2008.

Дане середовище розробки було обрано по декількох критеріях:

- Підтримка мови C#
- Інтуїтивно зрозумілий інтерфейс
- Можливості графічного редактору, підходящі до задач

Так як усі критерії задовільнено — середовищем розробки було обрано саме Visual Studio 2019.

2.2 Проектування структури системи

На рисунку 2.1 зображено блок - схему алгоритму роботи системи.

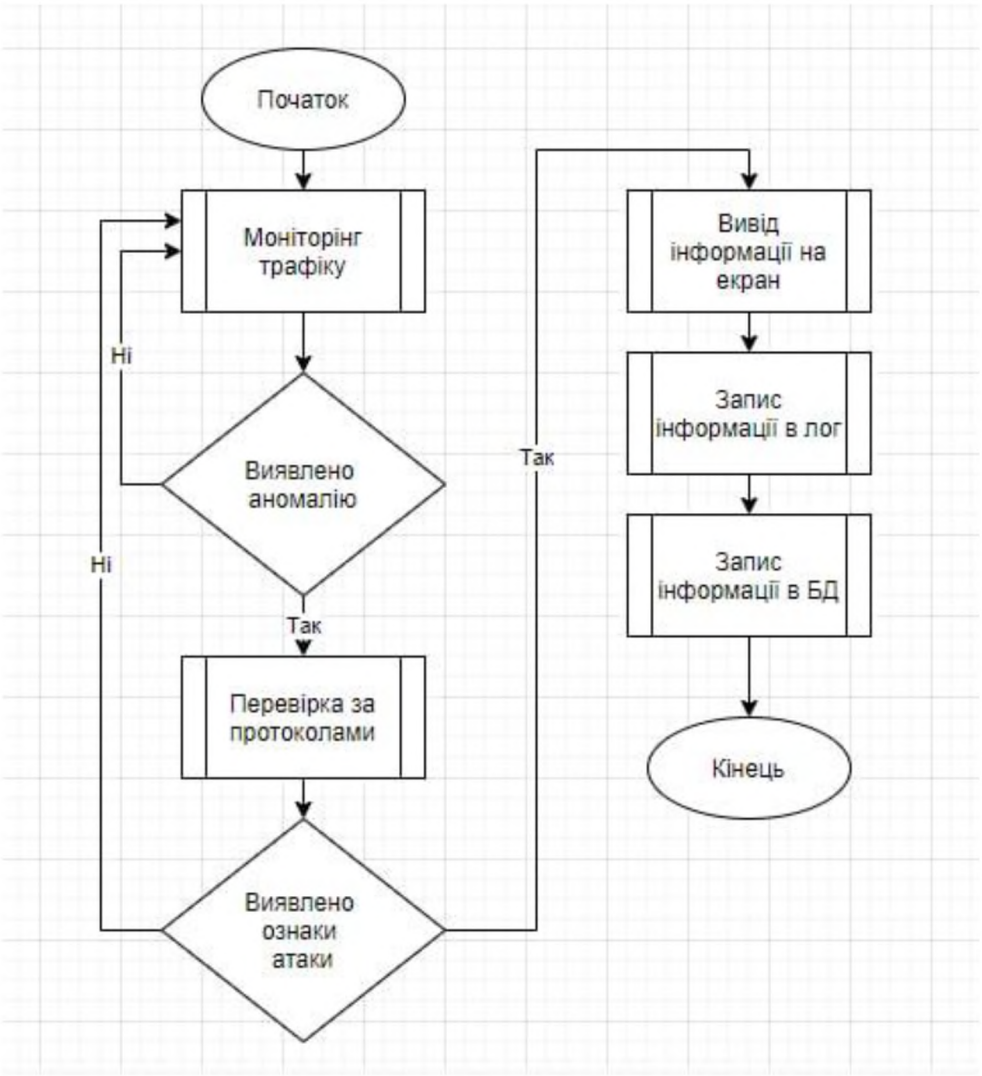


Рисунок 2.1 — Блок - схема роботи алгоритму

На рисунку 2.2 зображено загальну структурну схему системи.

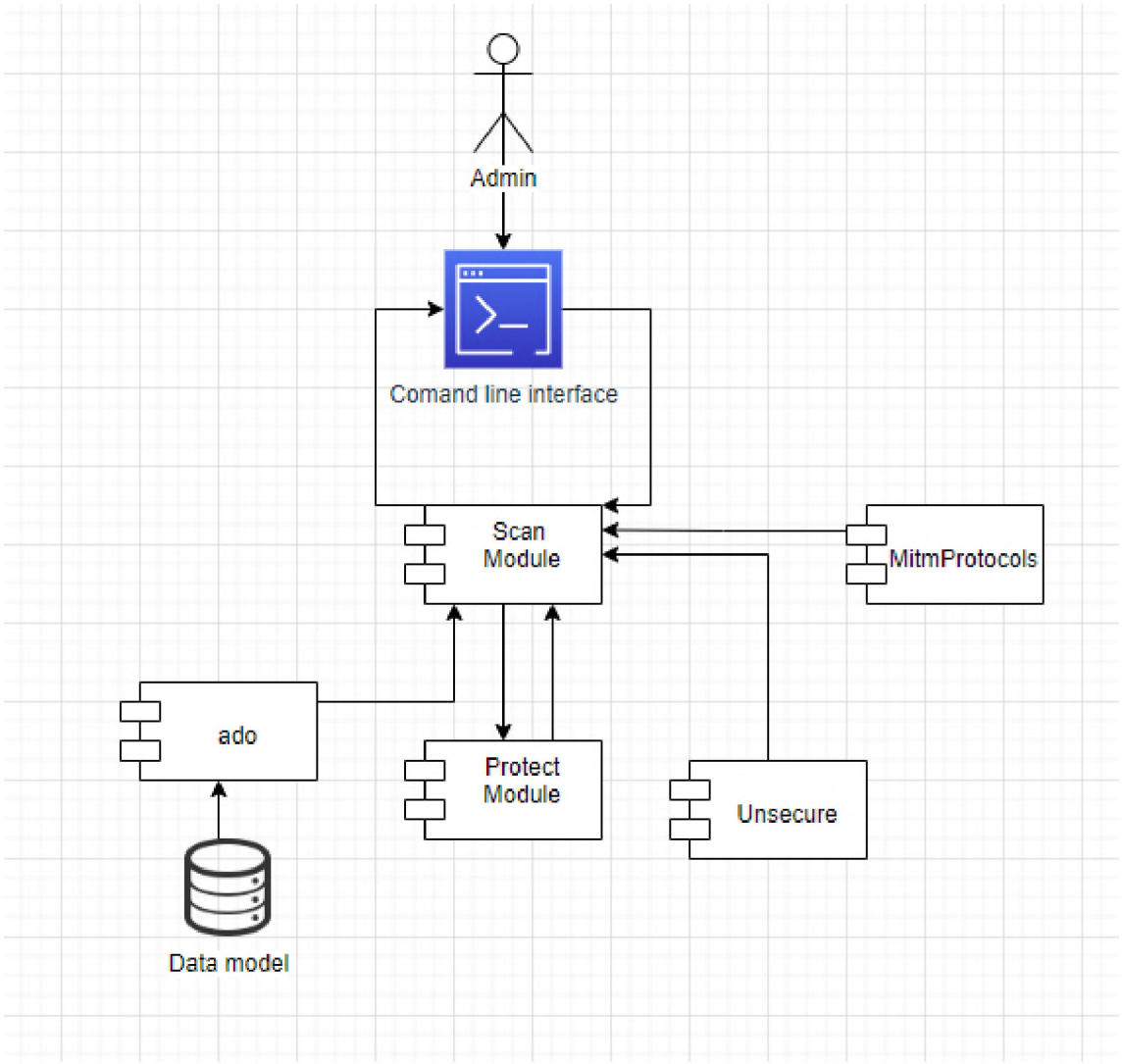


Рисунок 2.2 — Структурна схема

Структура системи виявлення вторгнень складається з консольного інтерфейсу користувача, за допомогою якого користувач вводить команди, які відповідають за подання запитів на модуль сканування вторгнень, який, в свою чергу взаємодіє з модулями протоколів, захисту, пошуку вразливостей та моделлю даних.

Кожен з модулів є окремою завершеною частиною загальної картини, яка в сукупності дає нормальне розподілення функціоналу системи між структурними частинами.

Модуль ado необхідний для взаємодії з БД, в якій зберігається інформація про порушення. Модуль протоколів зберігає в собі класифікаційні правила для виявлення атак.

Модулі захисту і сканування, фактично, є розподіленою реалізацією виявлення атаки.

2.3 Розробка внутрішньої будови системи

Розробка внутрішньої будови системи розуміє під собою створення системи класів. Розроблена система класів зображена на рисунку 2.3 у вигляді діаграми класів.

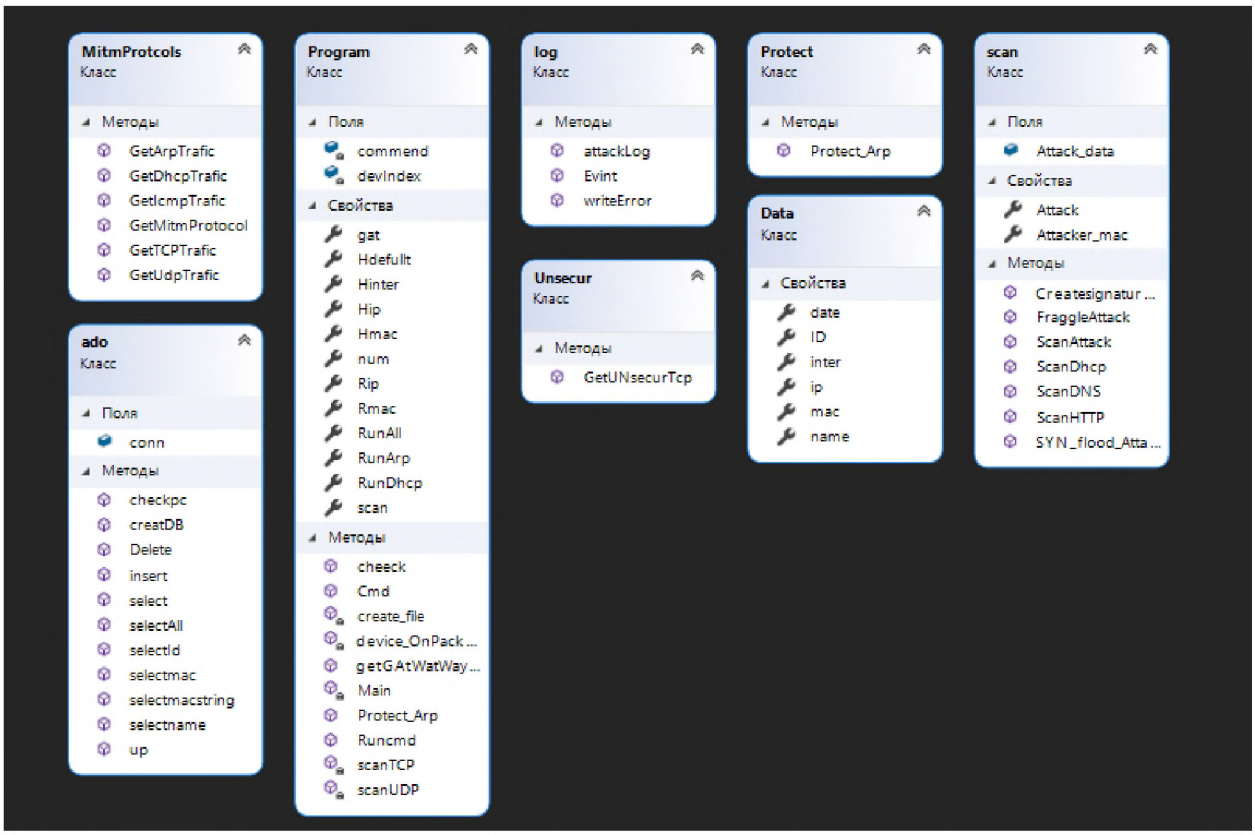


Рисунок 2.3 — Діаграма класів системи

- Клас MitmProtocols зберігає дані стосовно характеристик, за якими виявляється атака.
- Клас ado призначений для взаємодії з БД.
- Клас Data є конейнером для даних, шо записуються в базу даних.
- Клас log відповідає за логування роботи системи в рантаймі.
- Клас Scan відповідає за сканування системи.
- Клас Program є основним класом програми, який містить в собі обробку взаємодії ПЗ з користувачем і виклики всіх необхідних класів.

2.4 Розробка графічного інтерфейсу користувача

Графічний інтерфейс користувача представляє собою інтерфейс командного рядка.

Приклади інтерфейсу зображено на рисунках нижче.

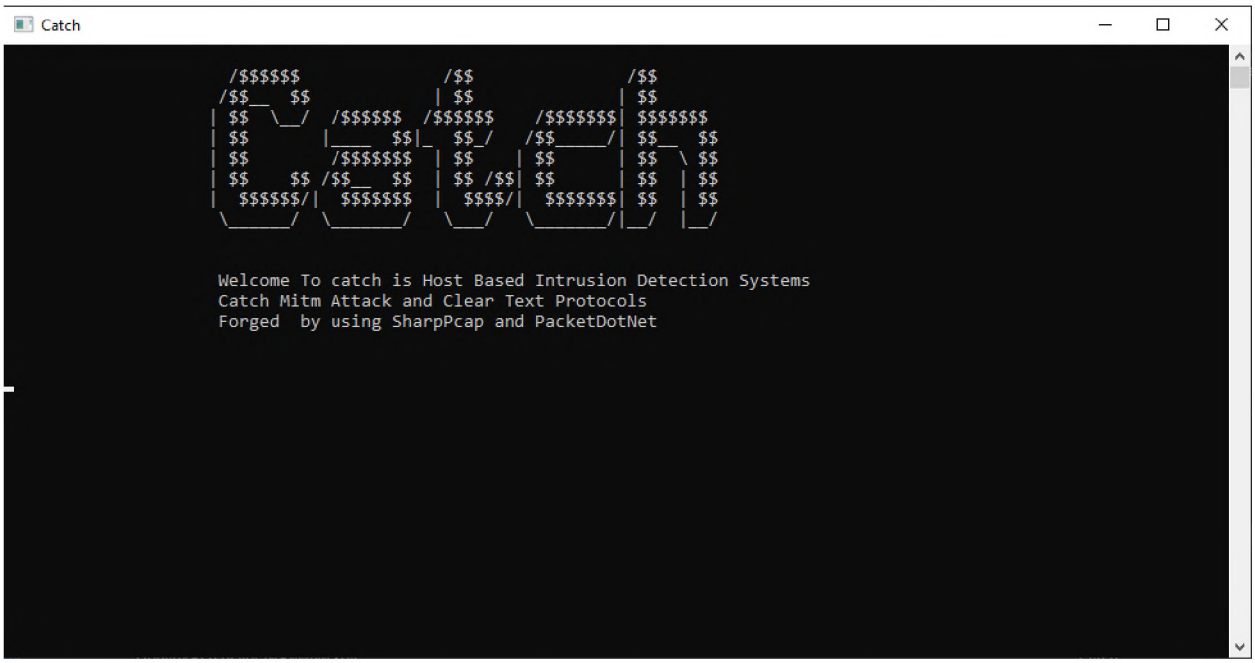


Рисунок 2.4 — Інтерфейс системи (1)

Стан інтерфейсу після запуску програми.

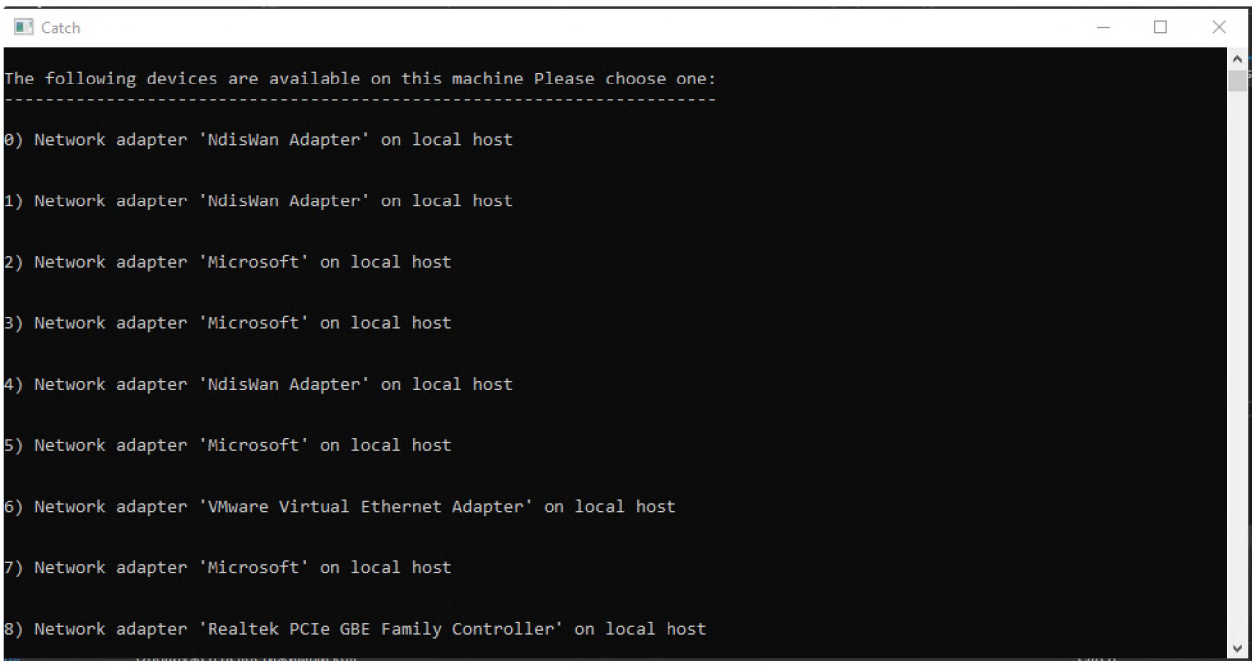


Рисунок 2.5 — Інтерфейс системи (2)

Стан інтерфейсу на момент сканування на предмет виявлення пристроїв, підключених до системи.

Рисунок 2.6 — Інтерфейс системи (3)

Стан інтерфейсу програмного модулю на момент вибору адаптеру для подальшого сканування трафіку на предмет виявлення загроз проникнення у систему.

2.5 Експериментальне дослідження

Для початку роботи на ком'ютері «зловмисника» слід встановити системи дистрибутив Kali Linux та завантажити Yersinia.

Рисунок 2.7 — Робочий стіл дистрибутиву Kali Linux

Далі слід запустити Yersinia і задати необхідні параметри сканування мережі.

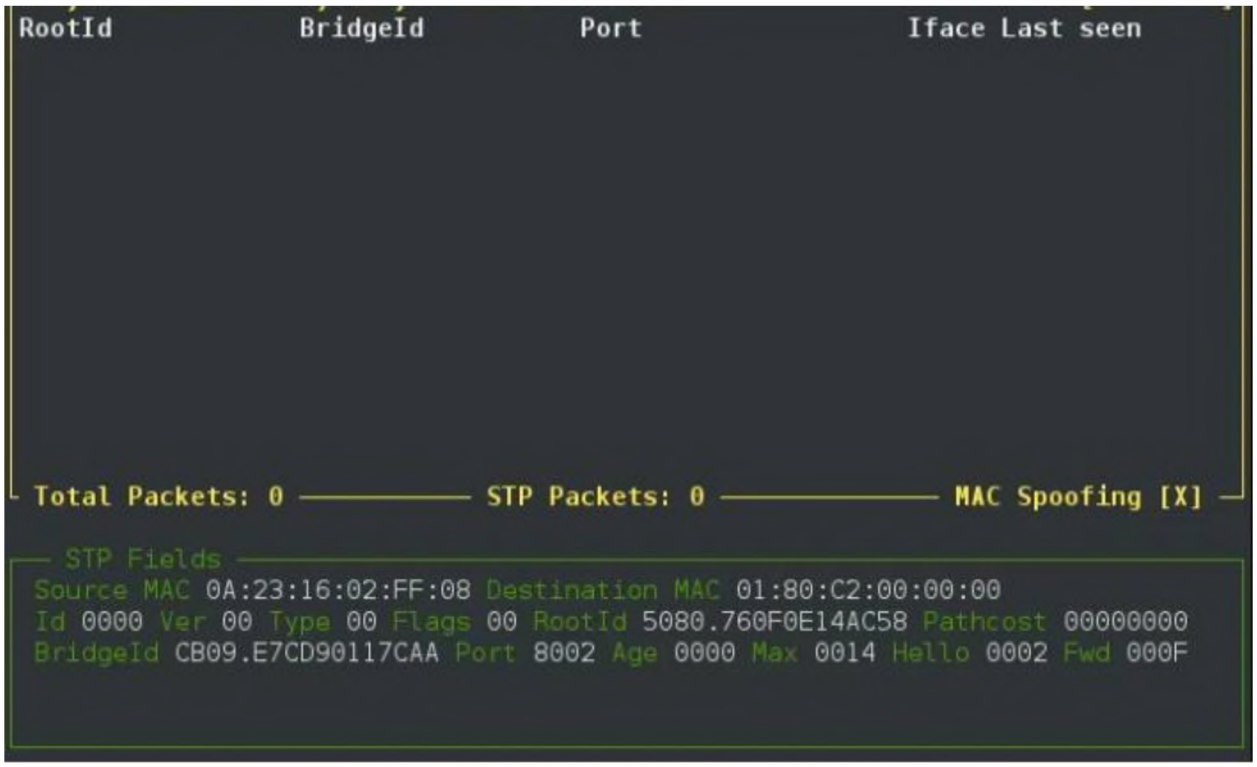


Рисунок 2.8— Yersinia, головне вікно

Далі слід обрати інтерфейс, який буде використовуватися для атаки, протокол та, безпосередньо атаку.

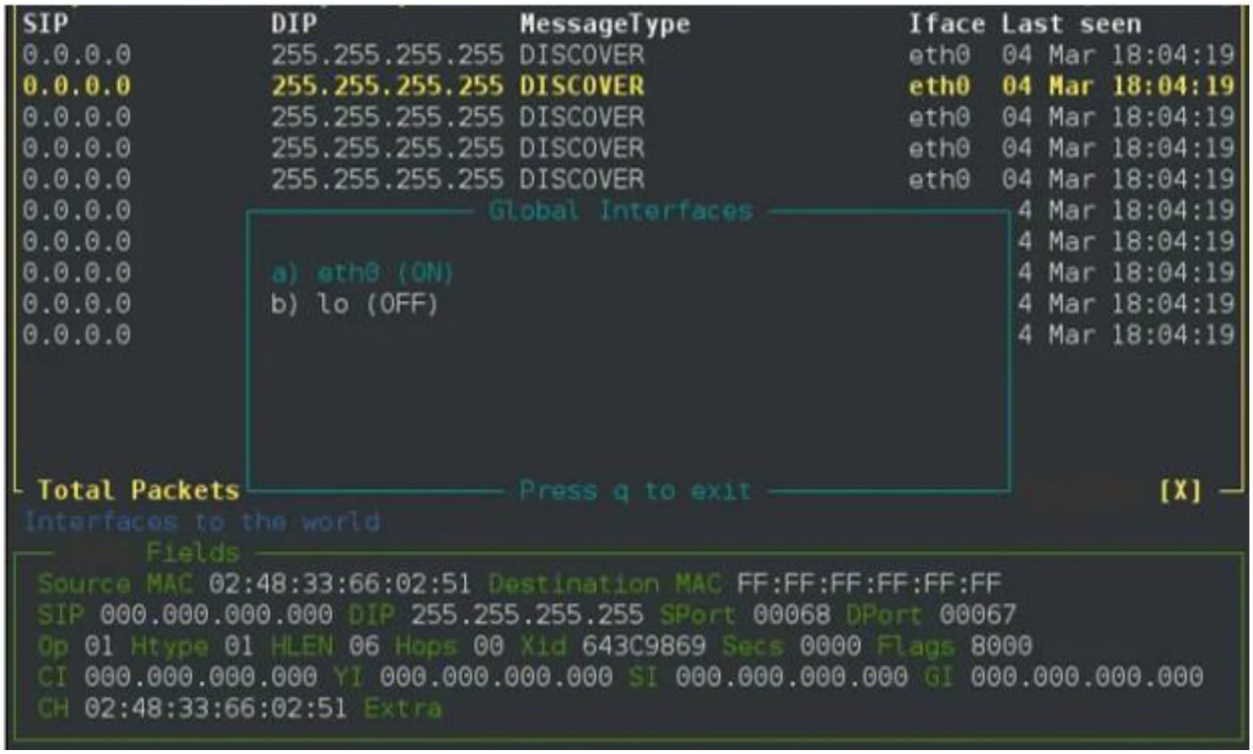


Рисунок 2.9 — Вибір інтерфейсу

Рисунок 2.10 — Вибір сервера

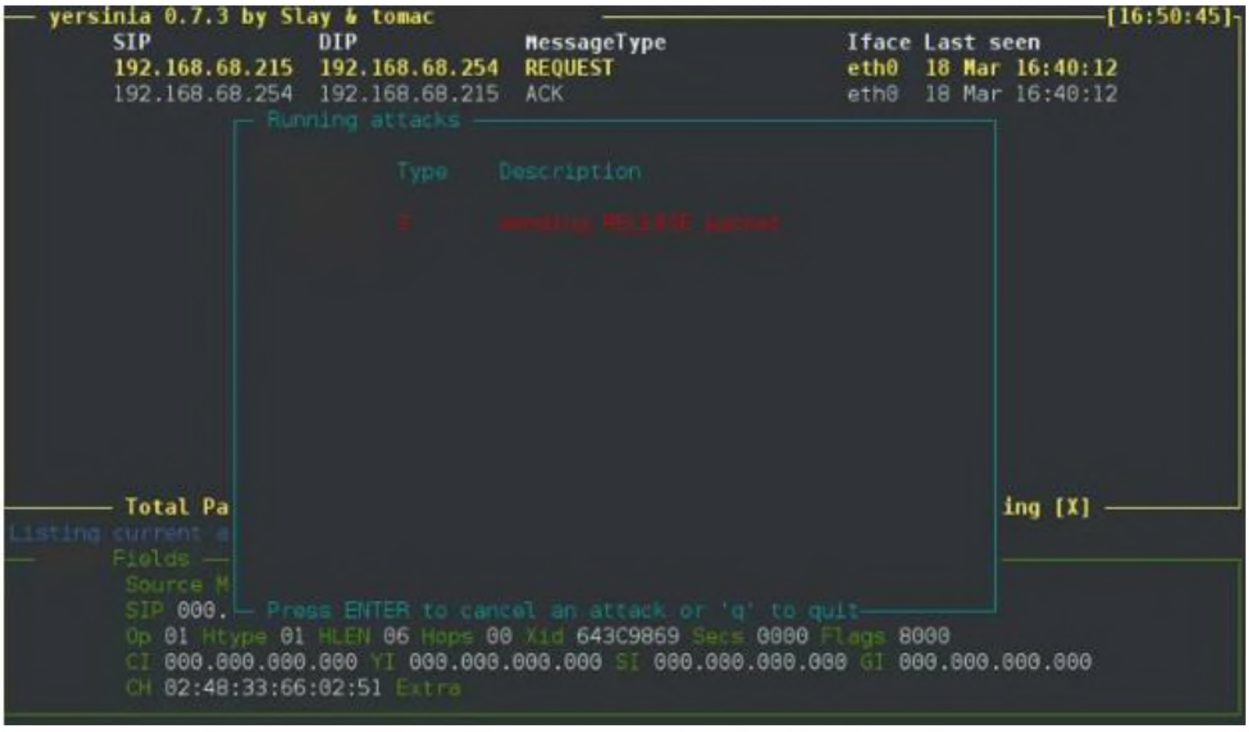


Рисунок 2.11 — Запущена атака

Після того, як буда запущена атака слід відкрити розроблену системи виявлення вторгнень на машині «жертви» та перевірити її реакцію на подібну ситуацію.

Послідовність дій буде представлена скріншотами з підписами на Рисунок 2.12-2.16.

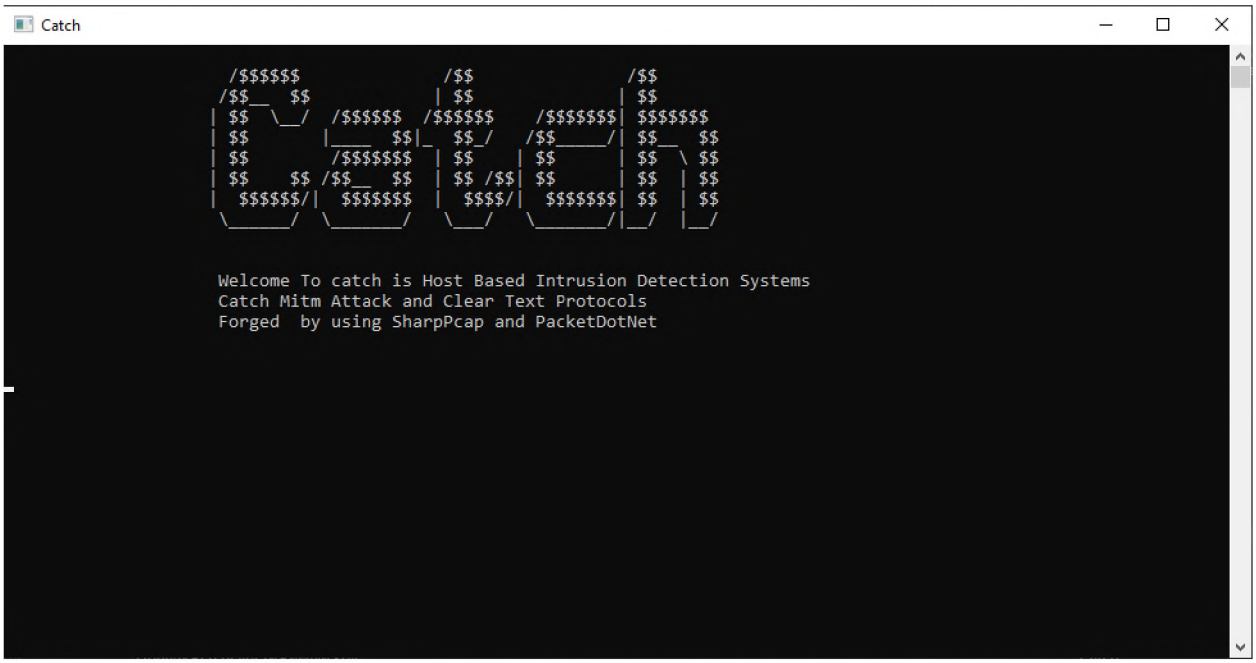


Рисунок 2.12 — Стан інтерфейсу після запуску програми

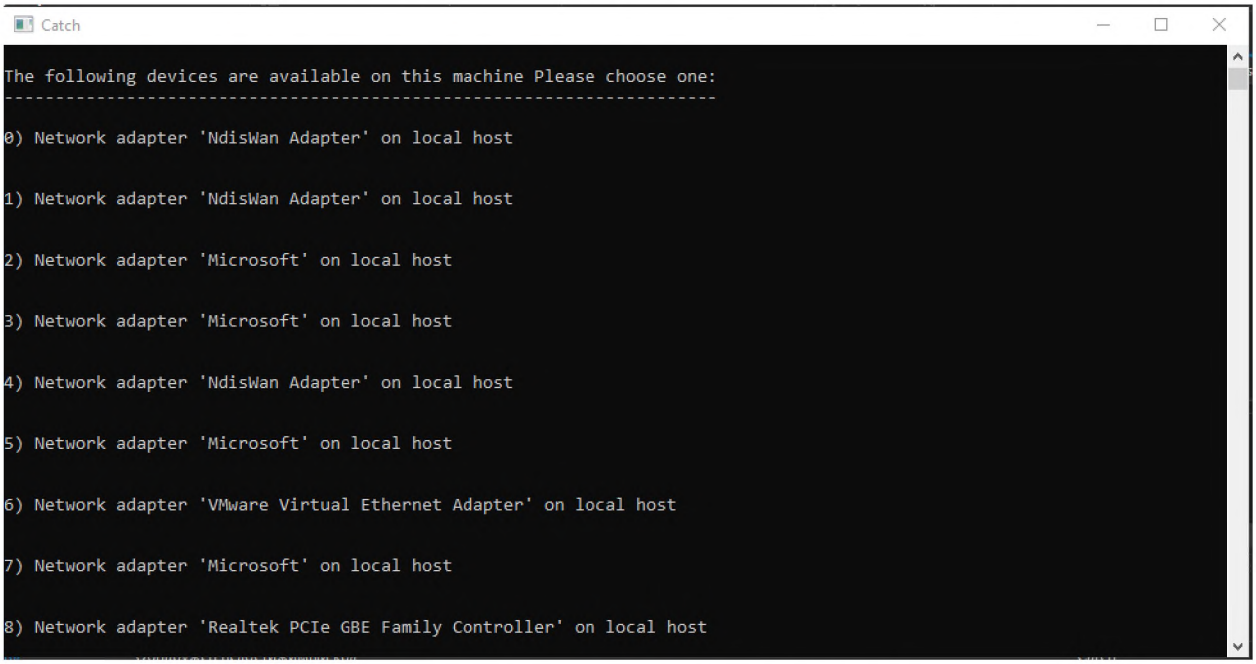


Рисунок 2.13 — Виявлені пристрої

Стан інтерфейсу на момент сканування на предмет виявлення пристроїв, підключених до системи.

Рисунок 2.14 — Вибір адаптеру

Стан інтерфейсу програмного модулю на момент вибору адаптеру для подальшого сканування трафіку на предмет виявлення загроз проникнення у систему.

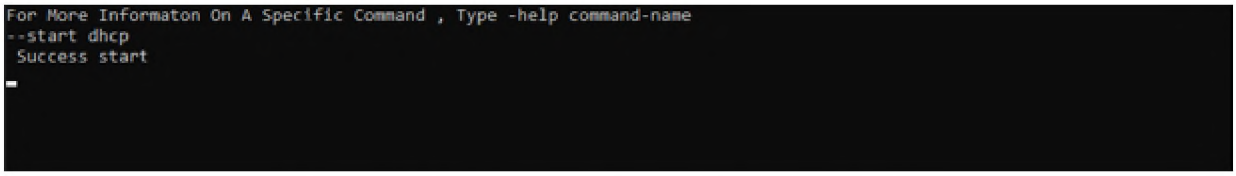


Рисунок 2.15 — Розпочато сканування

Рисунок 2.16 — Результат сканування

Як можна побачити на скріншоті, програма виявила атаку. Виходячи з результатів експерименти, можна зробити висновок, що програма повністю функціонує і здатна виявити вторгнення в систему.

Висновки до розділу 2

В ході виконання другого розділу було проведено аналітичний огляд інструментарію розробки, спроектовано структуру системи і її внутрішню будову.

Окрім цього проведено розробку графічного інтерфейсу користувача і експериментальне дослідження, націлене на перевірку працездатності і вірного функціонування розробленої системи.

В ході експериментального дослідження було виявлено, що програма реагує на атаку типу «людина посередині» за заздалегідь прописаним сценарієм, попереджаючи користувача при виявлені такої.

РОЗДІЛ 3
ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ

В даному розділі дипломної роботи проводиться економічне обґрунтування доцільності розробки програмного забезпечення. Зокрема, здійснюється розрахунок витрат на розробку програмного забезпечення, експлуатаційних витрат, ціни споживання проектного рішення. В заключній частині визначаються показники економічної ефективності нового програмного продукту, обґрунтовуються відповідні висновки.

Розроблене програмне забезпечення призначене для виявлення атак типу «людина посередині» на корпоративні мережі.

3.1 Розрахунок витрат на розробку програмного забезпечення

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

$$К = К_1+К_2 \qquad (3.1)$$

де $К_1$ - витрати на розробку програмних засобів, грн.

$К_2$ - витрати на відлагодження і дослідну експлуатацію програми рішення задачі на ЕОМ, грн.

- Витрати на розробку програмних засобів включають:
- витрати на оплату праці розробників;
- витрати на відрахування у спеціальні державні фонди (Вф,);
- витрати на куповані вироби (Кв);
- витрати на придбання спецобладнання для експериментальних робіт (Об);
- накладні витрати (Н);
- інші витрати (Ів).

Витрати на оплату праці розробників проекту визначаються за формулою:

$$З = \sum_{i=1}^{N} \sum_{j=1}^{M} n_{ij} t_{ij} C_{ij} \qquad (3.2)$$

де nij - чисельність розробників і-ої спеціальності j-го тарифного розряду, які приймають участь в проектуванні, чол.;

t_{ij} - час, який затрачений на розробку проекту співробітника і-ої спеціальності j-го тарифного розряду, днів;

C_{ij} - денна заробітна плата і-ої спеціальності j-го тарифного розряду, грн.;

$$C_{ij} = \frac{C_{ij}^{0}(1+h)}{p} \qquad (433)$$

де C_{ij} - основна місячна заробітна плата розробника і-ої спеціальності j-го тарифного розряду, грн.;

h - коефіцієнт, що визначає розмір додаткової заробітної плати;

p - середня кількість робочих днів у місяці (21).

Таблиця 3.1.

Вихідні дані для розрахунку витрат на оплату праці

№	Посада виконавців	Місячний оклад, грн.	Погодинна ставка, грн./година
1	Дизайнер	8500	52,7
2	Розробник	9000	55,9
3	Тестувальник	8500	52,7

Таблиця 3.2.

Розрахунок витрат на оплату праці

№	Спеціальність розробника	Час розробки,	Погодинна заробітна	Витрати на розробку, грн.
1	Дизайнер	18.5	52,7	974,95
2	Розробник	20	55,9	1118
3	Тестувальник	10	52,7	527
	Разом			2619,95

Величину відрахувань у спеціальні державні фонди визначають у процентному співвідношенні від суми основної та додаткової заробітної плати. Згідно діючого нормативного законодавства сума відрахувань у спеціальні державні фонди складає 22% від суми заробітної плати:

Вф=22: 100*З (3.4)

Вф=0,22*2619,95=576,38 грн.

При розробці даного програмного забезпечення спеціальне обладнання не використовувалось, тому витрати на спеціальне обладнання відсутні.

Накладні витрати проектних організацій включають три групи видатків: витрати на управління, загальногосподарські витрати, невиробничі витрати. Вони розраховуються за встановленими процентами до витрат на оплату праці:

$$H = \frac{30}{100} З \quad (3.5)$$

Н = 0,3*2619,95=785,98 грн.

Інші витрати відображають видатки, які не враховані в інших статтях витрат. Вони розраховуються за встановленими процентами до витрат на оплату праці:

$I_в$ =10:100*З (3.6)

Ів=0,1*2619,95=261,99 грн.

Витрати на розробку програмного забезпечення розраховуються за формулою:

К=З + Вф + Кв + Об + Н + Ів (3.7)

К =2619,95+576,38 +785,98+261,99=4244,30 грн.

Витрати на відлагодження і дослідну експлуатацію програмного забезпечення визначаються за формулою:

$K_2=S_{Мг}*t_{Від}$ (3.8)

де $S_{Мг}$ - вартість однієї машино-години роботи конкретного типу ЕОМ, грн./год.;

$t_{Від}$ - машинний час, витрачений на відлагодження і дослідну експлуатацію програмних засобів, год.

Загальна кількість днів роботи на ЕОМ рівна 60 днів. Середній щоденний час роботи на ЕОМ - 6 год., тому:

$t_{від}$=60*6=360 год.

За занальними даними для ЕОМ типу IBM PC/AT $S_{Мг}$ = 5 грн.

Отже:

K_2= 5,0*360 = 1800 грн.

Таблиця 3.4.

Кошторис витрат на розробку програмного забезпечення

№	Найменування елементів витрат	Сума витрат, грн.
1	Витрати на оплату праці	2619,95
2	Відрахування у спеціальні державні фонди	576,38
3	Накладні витрати	785,98
4	Інші витрати	261,99
5	Витрати на відлагодження і дослідну експлуатацію програмного забезпечення	1800
	Всього	6044,30

3.2 Визначення експлуатаційних витрат

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

Експлуатаційні одноразові витрати по програмному забезпеченню і аналогу включають вартість підготовки даних і вартість машино-годин роботи ЕОМ (за час дії програми):

$$E_{\text{П}} = E_{1\text{П}} + E_{2\text{П}} \qquad (3.12)$$

де $E_{\text{П}}$ - одноразові експлуатаційні витрати на проектне рішення (аналог), грн.;

$E_{1\text{П}}$ - вартість підготовки даних для експлуатації проектного рішення (аналогу), грн.;

$E_{2\text{П}}$ - вартість машино-годин роботи ЕОМ для виконання проектного рішення (аналогу), грн.

Річні експлуатаційні витрати $B_{\text{еП}}$ визначаються за формулою:

$$B_{E\text{П}} = E_{\text{П}} * N_{\text{П}} \qquad (3.13)$$

де $N_{\text{П}}$ - періодичність експлуатації проектного рішення (аналогу), раз/рік.

Вартість підготовки даних для роботи на ЕОМ визначається за формулою:

$$E_{1\text{П}} = \sum_{l=1}^{L} n_1 t_1 c_1 \qquad (3.14)$$

де 1 - номери категорій персоналу, який приймає участь у підготовці даних (1=1,2,...L);

n_1, - чисельність співробітників 1-ої категорії, чол.;

t_1, - трудоємність роботи співробітників 1-ої категорії по підготовці даних, год.;

c_1 — середнього динна ставка співробітника 1-ої категорії з врахуванням додаткової заробітної плати та відрахувань у спеціальні державні фонди, грн./год.

$$c_1 = \frac{c_1^0(1+b)}{m} \qquad (3.15)$$

де c_1^o - основна місячна заробітна плата працівника 1-ої категорії, грн.;

b - коефіцієнт, який враховує додаткову заробітну плату і відрахування у спеціальні державні фонди;

m - кількість робочих годин у місяці, год.

Для роботи з даними як для проектного рішення так і аналогу потрібен один працівник, основна місячна заробітна плата якого складає: c° = 6700 грн. Тоді, враховуючи додаткову заробітну плату:

$$c_1 = \frac{6700(1+0.57)}{162} = 64{,}9 \text{ грн/год}$$

Трудомісткість працівника по підготовці даних для проектного рішення складає 1 год., для аналога 1,5 год.

Таблиця 3.5.

Розрахунок витрат на підготовку даних та реалізацію проектного рішення на ЕОМ

№	Час роботи співробітників, год.	Середньогодинна заробітна плата,	Витрати , грн.
	Проектне рішення		
1	1	64,9	64,9
	Аналог		

1	1,5	64,9	97,4

Витрати на експлуатацію ЕОМ визначається за формулою:

$$E_{2П} = t * S_{МГ} \quad (3.16)$$

де t - витрати машинного часу для реалізації проектного рішення (аналогу), год.;

$S_{МГ}$ - вартість однієї машино-години роботи ЕОМ, грн./год.

$E_{2п} = 1 * 5,0 = 5,0$ грн.

$E_{2а} = 1,5 * 5,0 = 7,5$ грн.

Еп = 64,9 + 5,0 = 69,9 грн.

Еа = 97,4 + 7,5 = 104,9 грн.

$В_{еп} = 64,9 * 252 = 16354,8$ грн.

$В_{еа} = 97,4 * 252 = 24544,8$ грн.

3.3 Розрахунок ціни споживання проектного рішення

Ціна споживання - це витрати на придбання і експлуатацію проектного рішення за весь строк його служби:

$Цс_{п}=Ц_{п}+В_{епрv}$ (3.17)

де $Ц_{п}$ - ціна придбання проектного рішення, грн.:

$$Ц_{П} = K(1+\frac{П_{Р}}{100}) + K_{о} + K_{к} \quad (3.18)$$

де Пр - норматив рентабельності;

$К_{о}$ - витрати на прив'язку та освоєння проектного рішення на конкретному об'єкті, грн.;

$К_{к}$ - витрати на доукомплектування технічних засобів на об'єкті, грн.;

$Ц_{п}$=3527,38*(1+0,3) = 4585,6 грн.

$В_{епрv}$ - теперішня вартість витрат на експлуатацію проектного рішення (за весь час його експлуатації), грн.:

$$B_{enpv} = \sum_{t=0}^{T} \frac{B_{eП}}{(1+R)^{t}} \quad (3.19)$$

де $В_{еп}$ - річні експлуатаційні витрати, грн.;

Т - строк служби проектного рішення, років;

R - річна ставка проценту банківського

$$B_{enpv} = \sum_{t=1}^{5} \frac{16354,8}{(1+0.16)^{t}} = 55550.4 \text{ грн.}$$

$$B_{enpv} = \sum_{t=1}^{5} \frac{24544,8}{(1+0.16)^{t}} = 80366.8 \text{ грн.}$$

Вартість аналога становить 5000 грн., тому отримуємо:

$Ц_{сп}$ = 4585,6 + 55550,4 = 60136 грн.

$Ц_{са}$ = 5000 + 80366,8 = 85366.8 грн.

3.4 Розрахунок показників економічної ефективності

Економічний ефект в сфері проектування рішення:

$$Е_{пр}=Ц_{а}-Ц_{п} \quad (3.21)$$

$Е_{пр}$ = 5000,0 – 4585,6 = 414,4 грн.

Річний економічний ефект в сфері експлуатації:

$Е_{кс}=В_{еа}-В_{еп}$

$Е_{кс}$ = 24544,8 - 16354,8 = 8190 грн.

Додатковий економічний ефект у сфері експлуатації:

$$\Delta E_{екс} = \sum_{t=1}^{T} E_{екс}(1+R)^{T-t}$$

$$\Delta E_{екс} = \sum_{t=1}^{5} 8190 * (1+0{,}16)^{5-t} = 56323.7 \text{ грн.}$$

Сумарний ефект складає:

$$E = E_{пр} + \Delta E_{екс} = 414{,}4 + 56323.7 = 56738{,}1 \text{ грн}$$

Таблиця 3.6.

Показники економічної ефективності проектного рішення

№	Найменування	Одиниці вимірювання	Значення показників	
			Базовий варіант	Новий варіант
1	Капітальні вкладення	Грн.	-	3527,38
2	Ціна придбання	Грн.	5000,0	4585,6
3	Річні експлуатаційні витрати	Грн.	5922,0	3956,4
4	Ціна споживання	Грн.	24544,8	16354,8
5	Економічний ефект в сфері проектування	Грн.	-	414,4
6	Річний економічний ефект в сфері експлуатації	Грн.	-	8190
7	Додатковий ефект в сфері експлуатації	Грн.	-	56738,1
8	Сумарний ефект	Грн.	57152,5	

3.5 Рохрахунок можливих збитків

Для розрахунку вартості такого збитку можна застосувати наступну спрощену модель оцінки для умовного підприємства.

Необхідні данні для розрахунків:

P — умовний прибуток компанії в місяць, 500000 грн;

Prc — Середній % втрат при розкритті конфіденціальності даних, 60%.

Отже, можна підрахувати, що при відсутності виявлення вторгнень середнє значення місячних збитків компанії буде складати:

$$Зм = 500000*0,6 = 300000 \text{ грн}$$

Отже, середні збитки на рік при відсутності виявлення вторгнень складають:

$$Зр = Зм * 12 = 3600000 \text{ грн}$$

Висновки до розділу 3

В даному розділі проведено розрахунок витрат на розробку проектного рішення. Здійснено порівняння з існуючим аналогом, і цим показано, що дане проектне рішення має переваги в порівнянні з аналогами, зокрема: надійність, простота використання, гнучкість, зручність. Згідно проведеного економічного обґрунтування дане проектне рішення є конкурентноздатним. Крім того, отримано додатній економічний ефект у розмірі 56738,1грн. і тому розробка і впровадження цього проектного рішення є економічно доцільними.

ВИСНОВКИ

Мета роботи полягла в створенні власної програмної реалізації системи для виявлення атак типу «людина посередині» на корпоративні мережі.

В ході виконання роботи було виконано такі основні завдання:

- Проаналізовно існуючі види корпоративних ІТС, їх переваги та слабкості.
- На основі аналізу слабкостей корпоративної мережі типу “Зірка” розглянуто атаку “людина посередині”, та методи її реалізації.
- Проаналізовано існуючі рішення щодо виявлення вторгнень в мережу, з метою отримання загального розуміння принципів роботи подібних систем, їх переваг та недоліків.
- Розроблено структуру, алгоритм роботи та програмний засіб, який за рахунок сканування аномалій трафіку в корпоративній мережі типу “Зірка” може виявити вторгнення та сповістити про загрозу.
- Проведено експериментальне дослідження роботи програмного засобу для підтвердження ефективності його роботи з інсцінуванням атаки зловмисника на мережу з допомогою ОС Kali Linix та інструменту Yersinia.
- Наведено алгоритм інсцінування атаки на мережу та алгоритм використання програмного засобу для виявлення вторгнень із скриншотами станів інтерфейсу та налаштуваннями.
- Створено повноцінну реалізацію системи та проведено економічний аналіз доцільності даної розробки.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Schatz, Daniel; Bashroush, Rabih; Wall, Julie (2017). "Towards a More Representative Definition of Cyber Security". Journal of Digital Forensics, Security and Law. 12 (2). ISSN 1558-7215.
2. "Reliance spells end of road for ICT amateurs", 7 May 2013, The Australian
3. Kianpour, Mazaher; Kowalski, Stewart; Øverby, Harald (2021). "Systematically Understanding Cybersecurity Economics: A Survey". Sustainability. 13 (24): 13677. doi:10.3390/su132413677.
4. Stevens, Tim (11 June 2018). "Global Cybersecurity: New Directions in Theory and Methods" (PDF). Politics and Governance. 6 (2): 1–4. doi:10.17645/pag.v6i2.1569.
5. Misa, Thomas J. (2016). "Computer Security Discourse at RAND, SDC, and NSA (1958-1970)". IEEE Annals of the History of Computing. 38 (4): 12–25. doi:10.1109/MAHC.2016.48. S2CID 17609542.
6. J. Neumann, N. Statland and R. D. Webb (1977). "Post-processing audit tools and techniques" (PDF). US Department of Commerce, National Bureau of Standards. pp. 11-3–11-4.
7. Irwin, Luke (5 April 2018). "How NIST can protect the CIA triad, including the often overlooked 'I' – integrity". Retrieved 16 January 2021.
8. Perrin, Chad (30 June 2008). "The CIA Triad". Retrieved 31 May 2012.
9. Stoneburner, G.; Hayden, C.; Feringa, A. (2004). "Engineering Principles for Information Technology Security" (PDF). csrc.nist.gov. doi:10.6028/NIST.SP.800-27rA.
10. Yost, Jeffrey R. (April 2015). "The Origin and Early History of the Computer Security Software Products Industry". IEEE Annals of the History of Computing. 37 (2): 46–58. doi:10.1109/MAHC.2015.21. ISSN 1934-1547. S2CID 18929482.

11. Ellen Nakashima (26 January 2008). "Bush Order Expands Network Monitoring: Intelligence Agencies to Track Intrusions". The Washington Post. Retrieved 8 February 2021.
12. Nicole Perlroth (7 February 2021). "How the U.S. Lost to Hackers". The New York Times. Archived from the original on 28 December 2021. Retrieved 9 February 2021.
13. "Computer Security and Mobile Security Challenges". researchgate.net. 3 December 2015. Archived from the original on 12 October 2016. Retrieved 4 August 2016.
14. "Ghidra".
15. "Syzbot: Google Continuously Fuzzing The Linux Kernel".
16. "Distributed Denial of Service Attack". csa.gov.sg. Archived from the original on 6 August 2016. Retrieved 12 November 2014.
17. "Multi-Vector Attacks Demand Multi-Vector Protection". MSSP Alert. 24 July 2018.
18. Millman, Renee (15 December 2017). "New polymorphic malware evades three-quarters of AV scanners". SC Magazine UK.
19. "Identifying Phishing Attempts". Case. Archived from the original on 13 September 2015. Retrieved 4 July 2016.
20. "Phishers send fake invoices". Consumer Information. 23 February 2018. Retrieved 17 February 2020.
21. Eilam, Eldad (2005). Reversing: secrets of reverseengineering. John Wiley & Sons. ISBN 978-0-7645-7481-8.
22. Arcos Sergio. "Social Engineering" (PDF). Archived (PDF) from the original on 3 December 2013.
23. Scannell, Kara (24 February 2016). "CEO email scam costs companies $2bn". Financial Times. No. 25 February 2016. Archived from the original on 23 June 2016. Retrieved 7 May 2016.

24."Bucks leak tax info of players, employees as result of email scam". Associated Press. 20 May 2016. Archived from the original on 20 May 2016. Retrieved 20 May 2016.

25."What is Spoofing? – Definition from Techopedia". Archived from the original on 30 June 2016.

ДОДАТОК А. Відомість матеріалів кваліфікаційної роботи

№	Формат	Найменування	Кількість листів	Примітки
Документація				
1	А4	Реферат	2	
2	А4	Список умовних скорочень	1	
3	А4	Зміст	1	
4	А4	Вступ	1	
5	А4	Аналіз предметної області і постановка задачі	13	
6	А4	Проектування і створення власної реалізації	19	
7	А4	Економічне обґрунтування доцільності розробки	11	
8	А4	Висновки	1	
9	А4	Перелік посилань	3	
10	А4	Додаток А	1	
11	А4	Додаток Б	1	
12	А4	Додаток В	1	
13	А4	Додаток Г	1	

ДОДАТОК Б. Перелік документів на фізичному носії

1 Явтушенко_АО_125-18-2_пз.docx

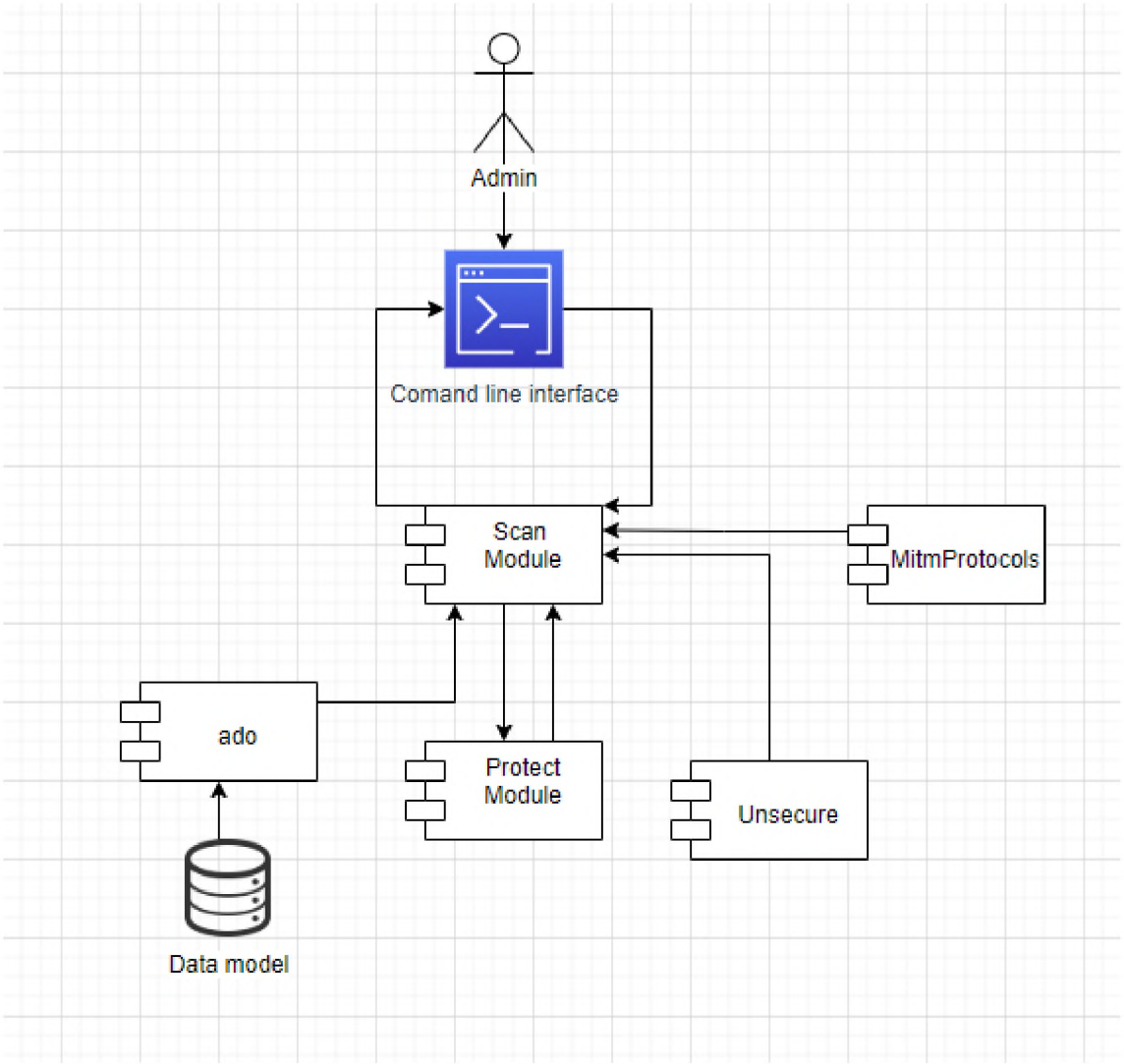
2 Явтушенко_АО_125-18-2_пз.pdf

3 Явтушенко_АО_125-18-2_дм.pptx

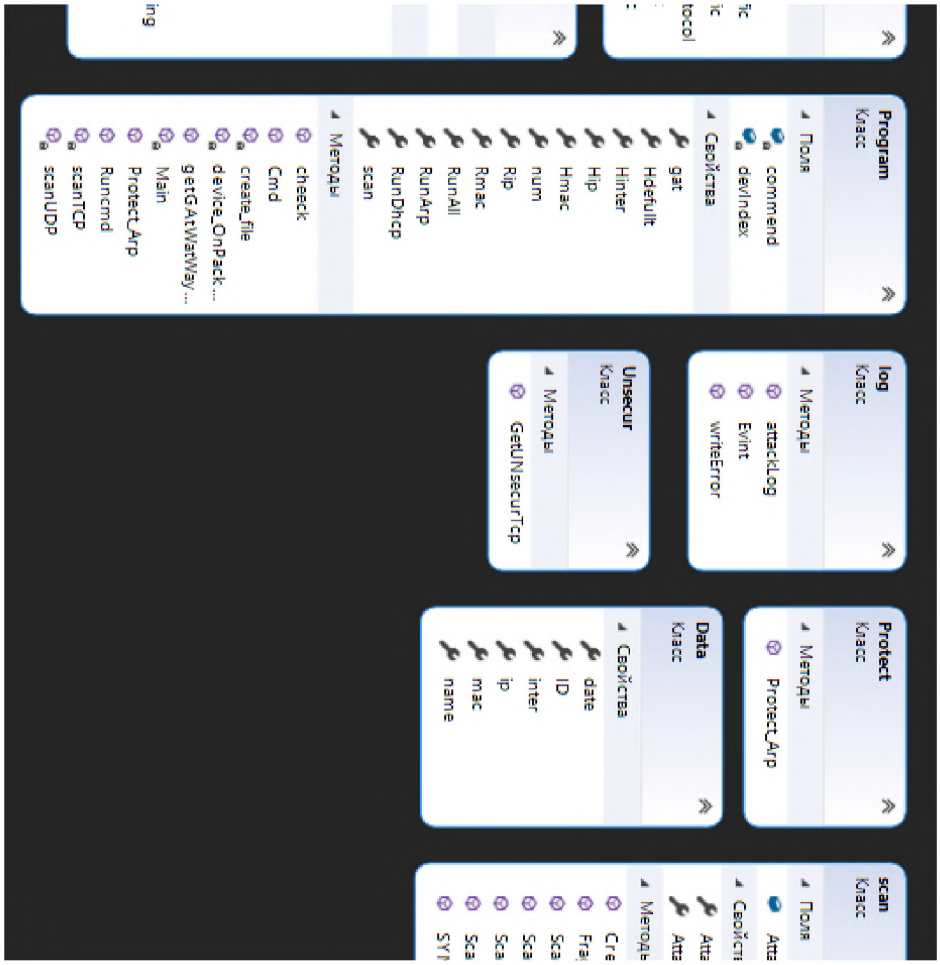
4 Явтушенко_АО_125-18-2_пз.pdf.p7s

5 Програма

ДОДАТОК В. Структурна схема системи



ДОДАТОК Г. Схема внутрішньої будови класів



ДОДАТОК Д. Відгук на кваліфікаційну роботу

студента групи 125-18-2

Явтушенко Артема Олексійовича

на тему

«Метод протидії атаці типу “людина посередині” в корпоративній ІТС»

Пояснювальна записка складається зі вступу, трьох розділів і висновків, викладених на ____ сторінках.

Метою кваліфікаційної роботи є розробка методу протидії атаці типу «людина посередині» в корпоративній ІТС та проведення розрахунків, щодо ефективності його використання.

Тема кваліфікаційної роботи безпосередньо пов'язана з об'єктом діяльності бакалавра спеціальності 125 «Кібербезпека». Для досягнення поставленої мети в кваліфікаційній роботі вирішуються наступні задачі: проаналізовано існуючі рішення щодо виявлення вторгнень в мережу, розроблено структуру, алгоритм роботи та проведено економічний аналіз доцільності даної розробки.

Практичне значення роботи полягає в тому, що запропонований програмний засіб дозволяє за рахунок сканування аномалій трафіку в мережі виявити вторгнення

До недоліків роботи слід віднести недостатню проработку структури програми та незначні невідповідності вимогам оформлення.

За час дипломування Явтушенко Артем Олексійович проявив себе фахівцем, здатним самостійно вирішувати поставлені задачі та заслуговує присвоєння кваліфікації бакалавра за спеціальністю 125 Кібербезпека, освітньо-професійна програма «Кібербезпека» .

Рівень запозичень у кваліфікаційній роботі не перевищує вимог “Положення про систему виявлення та запобігання плагіату”.

Кваліфікаційна робота заслуговує оцінки « 80 / добре ».

Керівник кваліфікаційної роботи к.т.н., доц. Герасіна О.В.

Керівник спеціальної частини ас. Мілінчук Ю.А.

ДОДАТОК Е. Відгук керівника економічного розділу

Економічний розділ виконаний відповідно до вимог, які ставляться до кваліфікаційних робіт, та заслуговує на оцінку 70 б. («задовільно»).

Керівник розділу _____ доц. Пілова Д.П.

(підпис) (ініціали, прізвище)