Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Галузь знань 12 – Інформаційні технології

Спеціальність 123 –Комп'ютерна інженерія

на тему «Програмно-технічний засіб виявлення вторгнень в IT-інфраструктури»

КвРКІ. 16039.20.01.20.16 ПЗ

Виконав: студент 2 курсу, група КІ2м-20-1 Сокальський Д.О.
Підпис Ініціали, прізвище

Керівник доктор техн. наук, професор Лисенко С.М.
Науковий ступінь, вчене звання Підпис Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КІІС, д.т.н., проф.
Т.О. Говорущенко
__ _____ 2022 р.

Хмельницький, 2022

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Освітній рівень магістр

Галузь знань 12 Інформаційні технології

Спеціальність 123Комп'ютерна інженерія

Освітня програма освітньо-наукова програма «Комп'ютерна інженерія та програмування»

ЗАТВЕРДЖУЮ

Зав. кафедри Т.О.Говорущенко

" 01 " 09 2021 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ)

Сокальському Дмитру Олександровичу

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Програмно-технічний засіб виявлення вторгнень в IT-інфраструктури

Керівник проекту (роботи) Лисенко С.М., д.т.н., професор

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 06.01.2022 р. №1

2. Строк подання студентом проекту (роботи) на кафедру 03.05.2022 р.

3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Дослідження систем виявлення вторгнень

Моделювання процесу виявлення вторгнень в іт-інфраструктури

Удосконалений метод виявлення вторгнень в іт-інфраструктури

Програмно-технічний засіб виявлення вторгнень в іт-інфраструктури

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Лисенко С.М., професор кафедри КІІС		
Антиплагіат	Нічепорук А.О., доцент кафедри КІІС		

7. Дата видачі завдання « 06 » 09 2021р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики ДРМ з керівником	05.09.2021	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об`єкта та предмета дослідження	05.10.2021	виконано
3	Робота над розділом 1 –аналіз відомих моделей, методів за темою; постановка задачі	05.11.2021	виконано
4	Робота над розділом 2 – розробка моделей для вирішення поставленої задачі	05.12.2021	виконано
5	Робота над науковою статтею	05.01.2022	виконано
6	Робота над розділом 3 – розробка методів для вирішення поставленої задачі	15.02.2022	виконано
7	Робота над розділом 4 – проектування та розробка ПЗ для вирішення поставленої задачі, експериментальна частина	05.04.2022	виконано
8	Оформлення пояснювальної записки згідно вимог	15.04.2022	виконано
9	Попередній захист ДРМ	18.04.2022	виконано
10	Захист ДРМ на засіданні ЕК	До 10.05.2022	

Студент Підпис Д.О. Сокальський Ініціали, прізвище

Керівник проекту (роботи) Підпис С.М. Лисенко Ініціали, прізвище

РЕФЕРАТ

Тема дипломної роботи: «Програмно-технічний засіб виявлення вторгнень в ІТ-інфраструктури».

Автор роботи: Сокальський Дмитро Олександрович

Керівник роботи: д.т.н., професор Лисенко С.М.

Пояснювальна записка: 93 с., 25 рис., 6 табл., 4 дод., 52 джерело.

СИСТЕМИ ВИЯВЛЕННЯ ВТОРГНЕНЬ, МОДЕЛЮВАННЯ DES, ЗАСТОСУВАННЯ І-ДЕТЕКТОРА, ,ПРОГРАМНО-ТЕХНІЧНИЙ ЗАСІБ ВИЯВЛЕННЯ ВТОРГНЕНЬ В ІТ-ІНФРАСТРУКТУРИ.

Об'єкт дослідження – процес виявлення вторгнень в ІТ-інфраструктури.

Предмет дослідження – моделі, удосконалений метод та програмно-технічні засоби виявлення вторгнень в ІТ-інфраструктури.

Методи дослідження. Для розв'язання поставлених задач використовуються основні положення системного аналізу, методів аналізу даних, теорії дискретної математики, дискретно-подійного моделювання, теорії комп'ютерних мереж та систем.

Наукова новизна отриманих результатів:

1.Удосконалено метод виявлення вторгнень в ІТ-інфраструктури, який на відміну від відомих ґрунтується на застосуванні DES-моделей, які характеризуються дискретним простором станів і певною динамікою, керованою подіями: в нормальних умовах, а також за кожного з умов атаки.

2. Набули подальшого розвитку програмно-технічні засоби виявлення вторгнень в ІТ-інфраструктури, які забезпечують виявлення вторгнень з високою достовірністю та ефективністю.

Практична цінність. В результаті виконаного наукового дослідження було розроблено програмно-технічних засобів виявлення вторгнень в ІТ-інфраструктури, які забезпечує виявлення вторгнень з високою достовірністю та ефективністю.

Дослідження, представлені у кваліфікаційній роботі, проводились в рамках держбюджетної НДР Хмельницького національного університету 1Б-2021 «Самоорганізована розподілена система виявлення зловмисного програмного забезпечення в комп'ютерних мережах» (ДР № 0121U109936) 2021-2022 рр..

За темою дипломної роботи опубліковано статтю на тему «Дослідження методів виявлення кібератак в комп'ютерних мережах як засобів до побудови резильєнтних до вторгнень ІТ-інфраструктур» в фаховому журналі «Комп'ютерні системи та інформаційні технології» №3 за 2021 рік [9], а також опубліковано тези у Всеукраїнській науково-практичній конференції Актуальні Проблеми Комп'ютерних Наук (АПКН-2021) [10]. Було взято участь у Всеукраїнській науково-практичній конференції Актуальні Проблеми Комп'ютерних Наук.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

IDS–Intrusion Detection System

DES – discrete-event simulation

КС– комп’ютерна система

ПЗ - програмне забезпечення

ЗПЗ– зловмисне програмне забезпечення

ВСТУП

Зі стрімким зростанням загроз безпеці IT-інфраструктурам, програмно-технічні засоби виявлення атак, які здатні відстежувати діяльність комп'ютерних мереж або окремих хостів на предмет зловмисної поведінки, є незамінним компонентом безпеки сучасних корпоративних мережі [1-3].

Найбільш поширеними програмно-технічними засобами виявлення атак є системи виявлення вторгнень (IDS) [2,4].

Серед найбільш поширених підходів до побудови системи виявлення вторгнень є такі методи проектування IDS: на основі підписів, які можуть виявляти лише відомі атаки, та системи на основі виявлення аномалій, які можуть виявляти як відомі, так і невідомі атаки, але генерують велику кількість помилкових спрацювань [5-7].

Існують такі класи атак, як атака на основі ARP, атака на основі ICMP, низькошвидкісна DoS-атака тощо, які уникають виявлення як за підписом, так і аномаліями [8-9]. Тому актуальною є задача розроблення програмно-технічних засобів виявлення вторгнень в IT-інфраструктури.

Сильна залежність від Інтернету та підключення по всьому світу значно збільшила потенційну шкоду, яку можуть завдати атаки, запущені через Інтернет на віддалені системи. Важко запобігти подібним атакам за допомогою політики безпеки, брандмауера чи інших механізмів, оскільки системне та прикладне програмне забезпечення завжди містить невідомі слабкі місця чи помилки. Крім того, зловмисники постійно експлуатують складні, часто непередбачені взаємодії між програмними компонентами та/або мережевими протоколами. Успішні атаки неминуче відбуваються, незважаючи на найкращі заходи безпеки. Однією з причин цього є стрімке зростання Інтернету та великої кількості мережевих систем, які існують у всіх типах організацій. Збільшення кількості мережевих машин призвело до зростання несанкціонованої діяльності [1], причому не тільки з боку зовнішніх зловмисників, але й із внутрішніх джерел, таких як незадоволені працівники та люди, які зловживають своїми привілеями для особистої вигоди [2]. Більше

того, через відсутність належної аутентифікації багатьох комунікаційних об’єктів в архітектурі шарів TCP/IP, кожен рівень схильний до різних загроз і вразливостей. Це вимагає наявності відповідного механізму системи виявлення для пошарового моніторингу можливих атак.

Адміністраторами безпеки в мережі використовується ряд компонентів безпеки, таких як брандмауер, проксі-сервер, список контролю доступу, система запобігання вторгненням (IPS) і система виявлення вторгнень (IDS). IDS стали важливим компонентом комп’ютерної безпеки для виявлення атак до того, як завдадуть масової шкоди, і в останні роки області виявлення вторгнень приділяється все більше уваги.

Вторгнення – це будь-який набір дій, які намагаються порушити конфіденційність, цілісність або доступність комп’ютерного ресурсу [3]. IDS – це пристрій або програмне забезпечення, яке відстежує діяльність мережі та/або системи на предмет зловмисної поведінки чи порушень політики та готує звіт для системного адміністратора. IDS в основному передбачає виявлення того, чи намагався якийсь суб’єкт недипломатичним способом отримати доступ до системних ресурсів чи гірше. IDS може бути заснований на хості або на основі мережі залежно від того, чи контролюється один хост чи мережу. IDS на основі хосту (HIDS) – це IDS, який відстежує та аналізує внутрішні елементи обчислювальної системи, наприклад, виконання програм, системні виклики тощо, щоб виявити шкідливу поведінку. Мережевий IDS (NIDS) – це IDS, який виявляє шкідливу поведінку шляхом моніторингу мережевого трафіку. Решта цієї тези в основному стосується NIDS і відтепер якщо не зазначено інше, термін IDS відноситься до NIDS.

Метою кваліфікаційної роботи магістра роботи є підвищення достовірності та ефективності виявлення вторгнень в IT-інфраструктури.

Поставлена мета досягається розв’язанням наступних задач:

1. дослідити особливості функціонування програмно-технічних засобів систем виявлення вторгнень в IT-інфраструктури;

2. проаналізувати сучасні програмно-технічних засоби виявлення вторгнень в IT-інфраструктури;

3. розробити модель процесу виявлення вторгнень в IT-інфраструктури

4. розробити модель функціонування програмно-технічних засобів виявлення вторгнень в IT-інфраструктури, виконати моделювання обробки підроблених пакетів при атаці на IT-інфраструктуру, побудувати модель атак на IT-інфраструктуру з вимірюваністю, побудувати модель атак на IT-інфраструктуру з керованістю;

5. розробити удосконалений метод виявлення вторгнень в IT-інфраструктури, що ґрунтується на застосуванні DES-моделей;

6. реалізувати програмно-технічні засоби виявлення вторгнень в IT-інфраструктури.

Об'єкт дослідження – процес виявлення вторгнень в IT-інфраструктури.

Предмет дослідження – моделі, удосконалений метод та програмно-технічні засоби виявлення вторгнень в IT-інфраструктури.

Методи дослідження. Для розв’язання поставлених задач використовуються основні положення системного аналізу, методів аналізу даних, теорії дискретної математики, дискретно-подійного моделювання, теорії комп’ютерних мереж та систем.

Наукова новизна отриманих результатів:

1.Удосконалено метод виявлення вторгнень в IT-інфраструктури, який на відміну від відомих ґрунтується на застосуванні DES-моделей, які характеризуються дискретним простором станів і певною динамікою, керованою подіями: в нормальних умовах, а також за кожного з умов атаки.

2. Набули подальшого розвитку програмно-технічні засоби виявлення вторгнень в IT-інфраструктури, які забезпечують виявлення вторгнень з високою достовірністю та ефективністю.

Практична цінність. В результаті виконаного наукового дослідження було розроблено програмно-технічних засобів виявлення вторгнень в IT-інфраструктури, які забезпечує виявлення вторгнень з високою достовірністю та ефективністю.

Зв’язок роботи з науковими програмами, планами, темами. Дослідження, представлені у кваліфікаційній роботі, проводились в рамках держбюджетної НДР Хмельницького національного університету 1Б-2021 «Самоорганізованарозподілена система виявлення зловмисного програмного забезпечення в комп’ютерних мережах» (ДР № 0121U109936) 2021-2022 рр..

За темою дипломної роботи опубліковано статтю на тему «Дослідження методів виявлення кібератак в комп’ютерних мережах як засобів до побудови резильєнтних до вторгнень ІТ-інфраструктур» в фаховому журналі «Комп’ютерні системи та інформаційні технології» №3 за 2021 рік [9], а також опубліковано тези у Всеукраїнській науково-практичній конференції Актуальні Проблеми Комп’ютерних Наук (АПКН-2021) [10]. Було взято участь у Всеукраїнській науково-практичній конференції Актуальні Проблеми Комп’ютерних Наук.

1 ДОСЛІДЖЕННЯ СИСТЕМ ВИЯВЛЕННЯ ВТОРГНЕНЬ

1.1 Таксономія IDS на основі виявлення атак

IDS можна розділити на два класи залежно від типу атак:

1) IDS на основі підпису;
2) IDS на основі аномалій.

1.1.1 IDS на основі підпису

IDS, який використовує сигнатури для виявлення атак, називається IDS на основі сигнатур. Сигнатура – це шаблон, який IDS шукає в мережевому трафікі для виявлення атак. Набір відомих сигнатур атак зберігається в базі даних, і діяльність мережі або системи порівнюється з цими сигнатурами. Якщо збіг знайдено, спрацьовує тривожний сигнал (в цій дипломній роботі використовується як «попередження»). У літературі запропоновано ряд систем виявлення підписів. Snort [4] і EMERALD [5] є одними з широко використовуваних систем виявлення підписів.

1.1.2 IDS на основі аномалій

IDS на основі сигнатур може виявляти лише атаки, синтаксис або поведінка яких відомі. Однак якщо атака нова або її синтаксис і поведінка невідомі, зазвичай використовується інший клас IDS, який називається IDS на основі аномалій. IDS на основі аномалій може виявляти як відомі, так і невідомі атаки. Моделює безпечну поведінку системи з профілем, і будь-яке відхилення від відомого профілю розглядається як вторгнення. Доброякісний профіль представляється як набір параметрів (які також називають функціями) і витягуються з мережевих пакетів. Наприклад, кількість пакетів TCP SYN, що спостерігаються протягом часового вікна, є особливістю. Огляд різних IDS на основі аномалій можна знайти в [6,7].

1.2 Обмеження існуючих IDS

Хоча IDS на основі сигнатур ефективний для виявлення атак без створення величезної кількості помилкових спрацювань, але може виявляти лише відомі атаки. Тому такі IDS повинні постійно оновлюватися сигнатурами нових атак. Багато детекторів сигнатур розроблено для використання чітко визначених сигнатур, які не дозволяють їм виявляти варіанти поширених атак.

Anomaly IDS може виявляти незвичайну поведінку і, таким чином, мати можливість виявляти симптоми атак без специфічних знань про деталі, а також може виробляти інформацію, яка, у свою чергу, може бути використана для визначення сигнатур для детекторів неправильного використання. Однак IDS викликає велику кількість помилкових спрацювань через непередбачувану поведінку користувачів і мереж. Крім того, для того, щоб охарактеризувати нормальні моделі поведінки, це непросте завдання.

Існує певний клас атак, які неможливо виявити за допомогою жодного з цих двох типів IDS. Деякі з них: «Інсайдерська атака на основі протоколу розв’язання адреси (ARP), «Атака через повідомлення протоколу керування повідомленнями Інтернету (ICMP)», «Атака на протокол керування передачею (TCP)» тощо, які детально описані далі.

1.2.1 Атаки на основі ARP

На канальному рівні комп’ютери використовують іншу адресу, відому як адресу керування доступом до медіа (MAC) або апаратну адресу. Щоб доставити пакет на правильний комп’ютер, адреса Інтернет-протоколу (IP) має бути зіставлена з якоюсь MAC-адресою. ARP відповідає за пошук MAC-адреси за IP-адресою. Канальний рівень даних використовує MAC-адресу машини призначення для відправки пакетів. Якщо хост, який надсилає пакети, не знає MAC-адресу хоста призначення, тоді надсилає широкомовний запит із запитом «Яка MAC-адреса відповідає IP-адресі». Хост, який має IP-адресу в

широкомовному повідомленні, надсилає відправнику одноадресне повідомлення-відповідь із зазначенням його MAC-адреси. Щоб зменшити кількість широкомовних запитів, кожна машина підтримує таблицю, яка називається кеш ARP, яка містить зіставлення між IP і MAC. Записи в кеші ARP можуть бути статичними або динамічними. У динамічному кеші записи стираються, коли стають старшими за попередньо визначений термін. Проблема з ARP полягає в тому, що це протокол без стану. Будь-який хост після отримання будь-якого повідомлення відповіді ARP оновить свій кеш, не перевіряючи, чи було раніше надіслано запит, відповідний відповіді. Ця відсутність громадянства є основною лазівкою безпеки в протоколі і дозволяє зловмисним хостам створювати користувацькі ARP-пакети з підробленими парами IP-MAC, що призводить до атак підробки ARP.

Атаку не можна виявити IDS на основі сигнатур, оскільки немає різниці між звичайними та підробленими ARP-пакетами. Щоб здійснити цю атаку, зловмиснику необхідно відправити дуже мало пакетів ARP, зазвичай один або два, залишаючи всі статистичні властивості мережевого трафіку недоторканими. Отже, при такій атаці в мережі не помічається жодної аномалії.

У літературі пропонується ряд пасивних рішень для виявлення, пом’якшення та запобігання таких атак. Найбільш надійний спосіб запобігти атакам ARP – це вручну призначити фіксований IP всім хостам і підтримувати статичні пари IP-MAC на всіх хостах [8]. Однак у динамічному середовищі це не практичне рішення. Також було запропоновано декілька рішень на основі апаратних засобів (комутаторів) [9] та програмних засобів [10, 11, 12], які є гнучкішими, ніж статичні пари IP-MAC. Ці схеми передбачають запам’ятовування пар IP-MAC, вперше отриманих з мережевого трафіку, і виконання відповідних дій (наприклад, блокування IP) при пасивному спостереженні за будь-якими змінами після цього. Проблема з цими підходами на основі пасивного спостереження полягає в тому, що якщо перший отриманий пакет сам має підроблену MAC-адресу, тоді вся схема виходить з ладу. Крім того, будь-яка справжня зміна в парі IP-MAC буде відхилена (наприклад, при отриманні сповіщення за допомогою Безоплатного запиту та

відповіді). IDS на основі сигнатур також застосовувалися для виявлення атак на основі ARP [13]. Основною проблемою IDS на основі сигнатур є велика кількість помилкових спрацювань [13]. Повідомлені результати показують, що хибнонегативні показники досягають 40%.

Також були запропоновані активні методи виявлення ARP-атак [14, 15], коли IDS активно посилає пакети запитів (активні проби) хостам на додаток до спостережень (наприклад, зміни пар IP-MAC). У [14] ведеться база даних відомих пар IP-MAC, і при виявленні зміни нова пара активно перевіряється шляхом надсилання запиту з пакетом TCP SYN на відповідний IP. Справжній хост відповість SYN/ACK або RST залежно від того, відкритий чи закритий відповідний порт. Хоча ця схема може підтвердити справжність пар IP-MAC, вона порушує архітектуру мережевих шарів. Активна схема виявлення атаки Man-in-The-Middle (MiTM) запропонована в [15]. Спочатку виявляються всі хости з IP-переадресацією. Після цього IDS атакує всі такі хости по черзі та отруює їхні кеші таким чином, що весь трафік, який пересилає зловмисник, досягає IDS (замість хоста хоче надіслати зловмисник із переадресацією IP). Таким чином, IDS може відрізнити справжніх зловмисників MiTM від усіх хостів за допомогою переадресації IP.

Хоча активні методи виявлення атак ARP перевершують у порівнянні з пасивними, однак, атаки включають додатковий трафік через зондування. Результати досвіду в [15] показали, що накладні витрати на додатковий трафік є незначними, враховуючи пропускну здатність сучасних локальних мереж (LAN).

З огляду можна стверджувати, що схема виявлення/попередження атак ARP повинна мати наступні функції:

1) не слід змінювати стандартний ARP або порушувати архітектуру мережевих шарів;
2) повинен генерувати мінімальний додатковий трафік в мережі;
3) не повинно вимагати виправлення всіх систем.

Виявлення та діагностика несправностей (FDD) системи дискретних подій (DES) була адаптована для IDS у [16, 17, 18]. Структура DES

характеризується дискретним простором станів і динамікою, керованою подіями. Парадигма DES широко використовується для виявлення та діагностики несправностей (FDD) через простоту моделювання та пов’язаних з ним алгоритмів [19]. Крім того, більшість систем можна моделювати як DES, використовуючи певний рівень абстракції. Основна ідея полягає в розробці моделі DES для системи в нормальних умовах, а також при кожному з умов відмови. Після цього розроблено засіб оцінки стану, який називається діагностом (або детектором, якщо потрібне лише виявлення несправності), який спостерігає за послідовністю подій, створених системою, щоб вирішити, чи відповідають стани, через які система проходить, нормальній чи несправній моделі DES. IDS на основі DES [16, 17, 18] в основному відображають атаки на збої, а діагностик реалізований як механізм IDS. Основна мотивація використання DES-фреймворку для IDS полягає в тому, що фреймворк дозволяє розробляти IDS без зміни залучених протоколів. Загалом, основними сферами застосування FDD DES є фізичні системи, такі як камери ядерної реакції, хімічні заводи тощо, конструкція яких не змінюється для виявлення відмов. Таким чином, DES-діагност може працювати без зміни конструкції системи.

У[17, 18] використано DES для виявлення атак ARP і проілюстрували, що відповідні IDS використовують пасивну структуру DES [20]. Передбачається, що зонди надсилаються якимось зовнішнім контролером, якого немає в моделі. Однак, як уже обговорювалося, для ARP-атак активні схеми [14, 15] перевершують пасивні [8, 11, 12]. Активне моделювання механізму виявлення атак для виявлення атак на основі ARP стає необхідним, щоб у моделі не було зовнішнього компонента для полегшення перевірки. Запропонована IDS [17,18] передбачає, що існує якась зовнішня система поза IDS, яка надсилає зонд, і IDS пасивно відстежує відповідь, а сама структура не активна.

У цьому досідженні пропонується IDS для ARP-атак, засновану на теорії FDD активного фреймворка DES. Контролер-контролер (IDS) отримує весь трафік завдяки дзеркальному відображенню портів і надсилає запити на перевірку справжності запитів і відповідей ARP. Діагностика пасивно відстежує

всі події, а саме: запити ARP, відповіді APR, відповіді на зонд тощо, щоб визначити, чи це атака, чи нормальна ситуація.

Нижче перераховано внесок запропонованої системи виявлення:

1. Активна IDS для ARP-атак була розроблена шляхом адаптації теорії FDD для активної DES-фреймворку [21]. Оскільки парадигма DES є системою переходу станів, вона страждає від проблеми вибуху стану. Класичні системи, якими обробляються DES, зазвичай мають стани в діапазоні тисяч, які можна обробляти. Однак у випадку мережевих протоколів діапазон параметрів дуже великий, наприклад, розмір домену IP-адреси може бути O(256.256.256.256). Тому при адаптації парадигми активного DES [21] для розробки IDS проблема вибуху стану вирішується за допомогою змінних моделі.

2. Активний IDS на основі DES має такі характерні особливості:

(a) Дотримується стандартного ARP і не порушує принципи структури мережевих шарів.

(b) Генерує мінімальний додатковий трафік в мережі, оскільки активне зондування виконується лише тоді, коли в трафікі видно неперевірену пару IP-MAC. Це досягається шляхом підтримки таблиць для аутентифікованих і підроблених пар IP-MAC.

(c) Будучи мережевим IDS, передбачає встановлення лише на одному хості в мережі.

(d) Оскільки це програмний підхід, тоді не вимагається встановлення будь-якого додаткового обладнання. Єдина вимога полягає в тому, що комутатор повинен мати функцію відображення портів.

(e) Висока швидкість виявлення і точність досягаються при розумних витратах ресурсів з точки зору пропускної здатності, використання ЦП і пам'яті.

1.3 Атака через повідомлення ICMP

ICMP використовується на мережевому рівні для відправки односторонніх інформаційних повідомлень для реалізації різних можливостей

звітування про помилки, зворотного зв’язку та тестування. ICMP використовується в Інтернеті для виконання функції ізоляції несправностей, яка являє собою групу дій, які виконують хости та маршрутизатори, щоб визначити, чи є якийсь збій мережі. У ICMP немає можливості автентифікації, що призводить до атак із використанням ICMP. Це може призвести до відмови в обслуговуванні (DoS) або дозволу зловмиснику перехопити пакети.

Повідомлення ICMP можна розділити на дві категорії залежно від їх роботи: повідомлення про помилки та інформаційні повідомлення. Повідомлення про помилки використовуються для повідомлення про різного роду помилки, що виникають під час доставки пакетів. Інформаційні повідомлення використовуються для діагностики, тестування та інших інформаційних цілей.

Нижче наведено приклади атак на основі інформаційних повідомлень ICMP.

Атаки DoS в основному використовують повідомлення ICMP «Exo-запит/відповідь» або «Реклама маршрутизатора». Зловмисник може підробити одне з цих повідомлень ICMP і надіслати його одному або обом хостам, що спілкуються, щоб перервати їхнє з’єднання. Повідомлення ICMP також можна використовувати для перехоплення пакетів за допомогою повідомлення ICMP «Redirect», яке зазвичай використовується шлюзами, коли хост помилково припустив, що пункт призначення знаходиться не в локальній мережі. Підробивши повідомлення ICMP «Redirect», зловмисник може викачувати трафік з хоста, рекламуючи свою власну IP-адресу як новий маршрутизатор у повідомленні ICMP «Redirect».

Нижче наведено деякі приклади атак на основі повідомлень про помилки ICMP.

Повідомлення DestinationUnreachable [22] використовуються для інформування хоста про те, що пункт призначення недоступний через певну причину. Ці помилки можуть бути згенеровані в результаті передачі TCP, UDP або іншої ICMP. Повідомлення можуть генеруватися або зі шлюзу, або з хоста. Різні атаки, пов’язані з цими повідомленнями: хост, порт, недоступний

протокол тощо. Ці повідомлення про помилки можуть бути підроблені зловмисником і можуть бути використані для припинення підключення хоста до цільової мережі, що призводить до DoS. Помилка HostUnreachable, один тип повідомлення про помилку недоступного призначення генерується граничним шлюзом або маршрутизатором іншої підмережі. HostUnreachable надсилається маршрутизатором, коли отримує дейтаграму, яку не може доставити або переслати хосту призначення. Зловмисник може підробити це повідомлення про помилку та припинити зв'язок джерела з хостом призначення.

Доступні різні механізми виявлення для виявлення атак, пов'язаних з кожним із цих повідомлень ICMP. Суть їх у наступному.

У системах виявлення на основі сигнатур, заснованих на наборах правил, пакети ICMP відкидаються, коли підписи збігаються. CiscoFirewall [23] має набір сигнатур для захисту внутрішньої мережі. Наприклад, перевіряє, чи пакети ICMP мають прапор «Більше фрагментів», встановлений на 1, і чи довжина в IP-заголовку перевищує 1024 байти. Аналогічно, інший підпис для захисту від DoS в маршрутизаторах Cisco обмежує кількість недосяжних пакетів ICMP до одного на 500 мс. У офіційному документі [24] пропонується блокувати/відкидати пакети, пов'язані з переадресацією ICMP і повідомленнями про недоступність ICMP, щоб уникнути атак DoS. Оскільки такі системи виявлення мають заздалегідь визначені правила або сигнатури для конкретної атаки, ці IDS не виявлять навіть незначні зміни в атаках. У [25] пропонується конфігурація маршрутизатора для блокування всіх вихідних пакетів, які вказують на вихідну адресу, яка не є частиною підмережі. Ці конфігурації маршрутизатора блокують важливі пакети ICMP, що не бажано. Кожне повідомлення ICMP має своє власне значення, тому його не слід відкидати наосліп. У [26] повідомлення ICMP traceback використовуються для вивчення шляху, який проходять пакети через Інтернет. За наявності достатньої кількості повідомлень про відстеження від маршрутизаторів на шляху трафіку можна визначити джерело та шлях підроблених пакетів. Але такі розширення для повідомлень ICMP вимагають змін у глобальній інфраструктурі маршрутизації, що непрактично. У [27] представлено безпечний механізм

виявлення маршрутизатора для хостів в IPv6. Можуть використовуватися криптографічні методи для додання автентичності повідомлень ICMP, але ці методимають дорогі обчислювальні витрати та високі вимоги до ресурсів зберігання.

Для цих атак написання підпису неможливе, оскільки синтаксис і послідовність мережевого трафіку в звичайних і скомпрометованих ситуаціях не відрізняються. Таким чином, неможливо виявити ці атаки за допомогою IDS на основі сигнатур. Крім того, ці атаки не призводять до істотного відхилення в характеристиках мережевого трафіку. Отже, IDS на основі аномалій також не може ефективно виявити ці атаки. Знову ж таки для цих атак генерують надзвичайно високі тривоги. Ці недоліки в існуючих методах спонукали запропонувати новий підхід для виявлення атак на основі ICMP.

Виявлено, що для атак на основі ARP систему можна діагностувати за допомогою активної платформи DES. Це означає, що якщо атака має місце, можна узгодити її виникнення шляхом аналізу послідовності пакетів протягом кінцевого проміжку часу, використовуючи керовану подію (активна подія, яка називається зонд). Але атаки ICMP можна виявити за певних мережевих умов і не виявити в інших мережевих умовах. У деяких випадках неможливо зробити висновок про те, що атака сталася протягом кінцевої затримки після того, як атака відбулася.

Наприклад, якщо спробуват переконатися в справжності повідомлення про помилку ICMP (скажімо, повідомлення про недосяжний пункт призначення) за допомогою активного зонда, отримана відповідь зонда може не повернутись, якщо в мережі є перевантаження. Таким чином, детектор цих атак не завжди закінчується в нормальному стані/атаці, що робить його недіагностованим. Оскільки перевантаження є неконтрольованою подією, активний фреймворк DES не можна застосувати для атаки на основі ICMP.

У [20] автори запропонували структуру під назвою I-diagnostability для часткової діагностики проблем. Тут визначаються деякі події індикатора, а діагностика несправності перевіряється лише на тих шляхах, де за несправністю слідує індикаторна подія. Шляхи, де немає індикаторних подій,

можуть містити деякі невизначені стани, що заважає діагностиці системи в цілому. Для атак, пов’язаних із ICMP, шляхи, де є проблеми, як-от перевантаження, не доступні індикаторні події. Таким чином, ця структура обмеженої діагностики [20] прийнята для моделювання та проектування IDS для атак, пов’язаних з ICMP.

Основний внесок системи:

1. IDS для IMCP-атак було розроблено шляхом адаптації теорії FDD про I-Diagnosability [20]. Нижче наведено зміни в I-Diagnosability, які зробили придатними для проектування IDS:

(a) Як і у випадку з активною структурою DES, I-Diagnosability також має проблему вибуху стану, яка вирішується шляхом доповнення структури змінними моделі.

(b) I-детектор, який використовується в структурі I-Diagnosability [20], містить багато надлишкових станів. Таким чином, був розроблений зменшений I-детектор (так званий RI-детектор) шляхом усунення зайвих станів, і було показано, що здатність виявлення несправностей не порушена.

2. IDS на основі DES для ICMP має схожі характерні особливості, як і у випадку з ARP, а саме: відсутність вимоги до зміни ICMP, мінімальні додаткові витрати на трафік, розумні накладні витрати на ресурси, відсутність вимоги виправлення окремих хостів тощо.

1.4 Повільні DoS-атаки

TCP – це протокол транспортного рівня, який забезпечує процес обробки надійної доставки потоку байтів. Крім того, забезпечує контроль потоку, уникнення перевантажень і контроль помилок. Майже всі прикладні протоколи, які хочуть обробити надійну доставку, використовують цей протокол транспортного рівня. TCP – це складний і надійний протокол, що складається з багатьох алгоритмів, який значною мірою відповідає своїй меті. Оскільки основною функцією TCP є забезпечення надійного процесу потоку байтів для обробки зв’язку, при розробці цього протоколу мало або майже не

враховувалося того факту, що алгоритми, що використовуються в TCP, можуть бути використані зловмисниками. Зловмисники використовували алгоритм TCP для запуску різних атак, таких як атака SYN flood, захоплення сеансу TCP, атака RST і FIN, а також атаки з використанням алгоритму уникнення перевантажень. Низькошвидкісні атаки, спрямовані на TCP, і викликані низькошвидкісні TCP-націлені атаки – це такі атаки, які використовують алгоритм уникнення перевантажень.

Атаку, спрямовану на низьку швидкість TCP, також звану норовливою атакою [28], було визначено Олександром Кузмановичем та Едвардом В. Найтлі у 2003 році. У цій атаці зловмисник використовує алгоритм уникнення перевантажень протоколу TCP. Зловмисники атакують, перевантажуючи мережу на короткий час; і згодом мовчить відносно довше. Коли мережа перевантажується до цільового сервера, інші справжні потоки TCP, призначені цьому серверу, втрачають усі або лише деякі свої сегменти. Це призводить до того, що справжні потоки TCP сповільнюють швидкість передачі. Відповідно до алгоритму запобігання перевантаженню TCP, щоразу, коли сегмент втрачено і його підтвердження не приходить до періоду очікування повторної передачі (RTO), швидкість передачі відправником різко знижується до фази повільного початку з вікном перевантаження 1 максимальний розмір сегмента (MSS). Як випливає з назви, під час виконання цих атак зловмисник підтримує низький середній рівень трафіку атаки, тому його не можна ідентифікувати за допомогою таких методів черги, як RED-PD [29].

Було запропоновано багато схем виявлення та захисту від цільових атак із низькою частотою TCP. У [30] запропоновано справедливе чергування та рандомізація рішення на основі RTO, що дає рівні шанси пакету кожного потоку в черзі маршрутизатора. Але планування справедливої черги не є масштабованим. Реалізація TCP кожного хоста повинна бути змінена, щоб рандомізувати RTO, а також це може спричинити погіршення продуктивності під час фази без атаки. У[31] запропонано рішення, в якому EdgeRouter відстежує потік трафіку, що йде від нього до цільового сервера, і використовує обмеження ресурсів для підозрюваного трафіку. Це може пошкодити законні

потоки TCP, оскільки для цієї атаки немає чіткого шаблону. Уу [32] запропоновано рішення, яке відзначає час прибуття кожного пакету та спостерігає різницю в часі пакетів для кожного потоку, щоб знайти характеристики землерийної атаки. Недоліком цього рішення є те, що воно має підтримувати інформацію для всіх потоків TCP, що вимагає більше ресурсів обробки та пам'яті в кожному маршрутизаторі. Також не може виявити атаку, коли вона виконується розподіленим способом. В [33] запропоновано схему, в якій підтримувалися дві шкали часу, одна для визначення довжини сплеску і друга для періоду атаки. Потоки відстежуються, щоб побачити, чи надходить трафік з більшою швидкістю. Цей метод не може виявити атаку, якщо вона виконується розподіленим способом, оскільки тривалість пакету або період атаки для окремого зловмисника недоступні. В [34] запропоновано схему, в якій вивчали вхідний і вихідний трафік TCP для виявлення атаки. Однак важко знайти оптимальний набір параметрів, які були б достатньо чутливими для виявлення розподіленої низькошвидкісної TCP-націленої атаки, зберігаючи при цьому низький рівень помилкових спрацьовувань. В [35] запропонованорішення, засноване на модифікації техніки скидання пакетів у чергу маршрутизатора шляхом класифікації потоків TCP за портами. Це рішення може зменшити інтенсивність атаки, але не може виявити та запобігти нападу. Крім того, він менш ефективний, якщо потік зловмисника має той самий клас, що й більшість потоків TCP, призначених цьому серверу.

З недоліків існуючих підходів видно, що потрібна схема, в якій рішення не потребує будь-якої модифікації формату пакетів TCP, не вимагає жодної настройки будь-якого проміжного маршрутизатора чи клієнтського хоста і не вимагає великих ресурсів.

Фреймворки DES, які обговорювалися дотепер, називають структурою на основі стану, оскільки нормальна та несправна поведінка системи, також звана специфікацією, моделюється системою переходу стану. Під час атаки, пов’язаної з ARP, звичайну послідовність подій і послідовність атаки можна розділити за допомогою активних зондів. Атаки ICMP можна діагностувати шляхом відокремлення нормальної та атакової послідовності подій у певних

шляхах за допомогою I-діагностики DES. Але у разі низькошвидкісної TCP DoS-атаки розділення звичайних подій і подій атаки неможливо ні за допомогою активної діагностики, ні за допомогою I-діагностики, оскільки атака та справжні сліди можуть не відрізнятися чітко, але можуть відрізнятися з певною ймовірністю.

Для виявлення цього типу атаки було адаптовано систему Stochastic DES [36]. DES на основі станів характеризується дискретним простором станів і динамікою, керованою подіями [19], але в стохастичному DES також вказуються ймовірності настання подій. Основна ідея полягає в розробці стохастичної моделі DES для системи в нормальних умовах і за кожного з несправних (або атаки) умов. Після цього розроблено стохастичний інструмент для оцінки стану, який називається стохастичний діагностик, який спостерігає за послідовністю подій, створених системою, щоб вирішити, чи відповідають стани, через які система проходить, нормальній моделі DES чи несправній моделі DES, і чи зменшується ймовірність першого. з часом напад вважається виявленим.

Внески цієї схеми перераховані нижче

1. IDS на основі стохастичної DES-фреймворка було розроблено для боротьби з низькошвидкісними TCP-націленими DoS-атаками. Щоб мати справу з дуже великим розміром доменів змінних, в адаптовану модель DES включені змінні стану.

2. IDS на основі стохастичної DES має мінімальні накладні витрати в звичайному випадку і покращує продуктивність у ситуації атаки, зводячи нанівець ефект атаки.

1.5 Атака на основі NDP

IPv6 використовує протокол NetworkDiscoveryProtocol (NDP), щоб знайти MAC-адресу. Традиційні атаки для використання ARP також є актуальними для NDP, оскільки не має громадянства та не має аутентифікації своїх повідомлень за замовчуванням. СпуфінгNeighborSolicitation (NS),

спуфингNeighborAdvertisement (NA), спуфінгRouterSolicitation (RS), підробка реклами маршрутизатора (RA), виявлення недоступності сусіда (NUD), виявлення дублікатів адреси (DAD), атака перенаправлення маршрутизатора тощо. приклади атак, пов'язаних з NDP. Хоча існують різні механізми виявлення та запобігання атак для ARP-атак (в IPv4, про які йшлося раніше), механізми ще не реалізовані в NDP (IPv6), оскільки протокол є відносно новим і починає використовуватися. Було запропоновано кілька механізмів для виявлення/попередження цих атак, але ці механізми або дорогі з точки зору обчислень, або вимагають керування криптографічними ключами, або передбачають зміну самого NDP.

Спуфінг NS і NA в NDP подібний до підміни запитів і відповідей в ARP. Крім того, існують додаткові атаки DAD, які також не виявляються як сигнатурними, так і аномальними IDS з аналогічної причини. Як і ARP, інформація про спарювання IP-MAC з пакетів NS і NA приймається без будь-якої перевірки, зловмисники можуть легко підробити пакет NS і NA за допомогою фальсифікованих пар IP-MAC. Підробка NS/NA передбачає, що зловмисний вузол надсилає повідомлення NS/NA цільовому вузлу, який має фальсифіковані пари IP-MAC. Це призводить до переспрямування всіх кадрів канального рівня на підроблену MAC-адресу.

У IPv6, коли новий вузол хоче з’явитися в мережі, перед використанням IP-адреси перевіряє, чи є інший вузол, який використовує ту саму IP-адресу, або претендує на ту саму адресу; це називається спробою DAD. Якщо такий вузол не знайдено, новий вузол може використовувати IP-адресу. В іншому випадку спробуєте використовувати іншу альтернативну IP-адресу або налаштувати статичну IP-адресу [37]. Концепція, згідно з якою відповідь вузла про те, що «він використовує ту саму IP-адресу або претендує на ту саму адресу», не перевірена, може використовуватися зловмисником для запуску DoS-атаки. Якщо зловмисник відповідає на кожну спробу DAD, зроблену хостом, який входить, то хост ніколи не зможе отримати адресу. Зловмисник може отримати адресу двома способами:

(1) Зловмисник може або відповісти пакетом NS, імітуючи, що також виконується DAD для тієї ж адреси.

(2) Зловмисник може відповісти пакетом NA, імітуючи, що уже використовує адресу. Було запропоновано різні запобіжні механізми для атак на основі NDP, які згадуються нижче. IPsec AH [38] може використовуватися з повідомленнями NDP для підвищення безпеки та перевірки через AH, що NA містить правильну та точну інформацію. Асоціації безпеки (SA) можна створити лише за допомогою обміну ключами в Інтернеті (IKE)[39]. Але для роботи IKE потрібен функціональний стек IP, і це призводить до проблеми з завантаженням. Таким чином, SA можна налаштувати лише вручну, що є втомливим або непрактичним завданням, враховуючи обсяг. Навіть якщо SA створено, неможливо перевірити право власності на динамічно згенеровані IP-адреси. SecureNeighborDiscovery (SEND) [40] визначає цей механізм, який використовує PKI для підписання повідомлень NDP. Крім того, управління ключами в середовищі локальної мережі є складною і громіздкою роботою для невеликої або середньої організації. Крім того, криптографічно згенеровані адреси (CGA)[41] використовуються, щоб уникнути спуфінгу в локальній мережі. Однак цей протокол ще не широко реалізований, і пов’язані з ним накладні витрати можуть викликати самі умови DoS. NeighborDiscoveryProtocolMonitor (NDPmon)[42], це інструмент, який працює з пакетами ICMPv6, спостерігає за локальною мережею, щоб перевірити, чи правильно поводяться вузли, які використовують повідомлення виявлення сусідів. Коли виявляється підозріле повідомлення NeighborDiscovery, повідомляє адміністратора, записуючи його в системному журналі. Проблема такого підходу полягає в тому, що якщо перший надісланий пакет сам має підроблену MAC-адресу, то вся система виходить з ладу. Крім того, будь-які справжні зміни в парі IP-MAC будуть відхилені.

Атаки, пов’язані з NDP, подібні до атак на основі ARP, а також не мають чітких ознак або аномалій. Тому для виявлення цих атак можна застосувати активну структуру DES [21]. Можна зазначити, що у всіх фреймворках DES, які обговорювалися раніше, моделювання нормальних умов і умов атаки

виконується вручну, що вважається правильним. Ефективність IDS на основі DES залежить від моделювання DES. Специфікація NDP і пов'язаних з ними атак є більш складною в порівнянні з їх аналогом в ARP. Отже, якщо для розробки IDS для атак NDP використовується активний фреймворк DES, є ймовірність компромісу в ефективності через крок моделювання вручну.

Для вирішення проблеми ручного моделювання в структурі DES на основі стану [20, 21]. В [43] запропонована парадигма DES на основі лінійної часової логіки (LTL). Як і у випадку з моделями на основі станів, у фреймворку DES на основі LTL також спочатку система в звичайних умовах і в умовах атаки розробляється вручну. Після цього звичайна специфікація системи виражається у вигляді формул LTL. Формули LTL потім перетворюються на автомати кінцевих станів (FSA) під назвою BuchiAutomata за допомогою автоматизованих інструментів [44]. Хоча моделювання та перетворення специфікацій у формули LTL виконуються вручну, автоматизовані методи, такі як перевірка моделі, використовуються для перевірки специфікацій LTL щодо моделі системи [66]. Правильність формул LTL і моделі системи можна визначити перед переходом до проектування детектора. Детектор у парадигмі DES на основі LTL отримано шляхом синхронізації пропозицій автоматів Бучі та моделі системи. Детектор спостерігає за системними слідами і виявляє несправність, якщо слід порушує формулу LTL.

Тому для проектування IDS для атак NDP використовується DES-фреймворк на основі LTL. Першим кроком є моделювання мережі в нормальному стані та наступних умовах атаки, виражаючи нормальну поведінку NDP як формулу LTL. На наступному кроці безпомилкова поведінка NDP представляється за допомогою BuchiAutomata, яка отримується автоматично зі специфікації LTL. Далі перевіряється правильність специфікації LTL та моделі системи. Нарешті, отриманий IDS на основі детектора DES, який перевіряється для всіх слідів, які порушують нормальну поведінку NDP.

1.6 Висновки та постановка задачі

В розділі було досліджено системи виявлення вторгнень. Також було проаналізовано таксономію IDS на основі виявлення атак, зокрема IDS на основі підпису та IDS на основі аномалій. було проаналізовано обмеження існуючих IDS.

В розділі також досліджено основні аспекти функціонування вторгнень засобами атаки на основі ARP, атаки через повідомлення ICMP, засобами повільних DoS-атак, а також атак на основі NDP.

Виявлено недоліки відомих методів програмно-технічних засобів систем виявлення вторгнень в IT-інфраструктури.

Зроблено висновок, щодо необхідності удосконалення методу виявлення вторгнень в IT-інфраструктури.

Таким чином, необхідно розв’язати наступні задачі:

1. дослідити особливості функціонування програмно-технічних засобів систем виявлення вторгнень в IT-інфраструктури;
2. проаналізувати сучасні програмно-технічних засоби виявлення вторгнень в IT-інфраструктури;
3. розробити модель процесу виявлення вторгнень в IT-інфраструктури
4. розробити модель функціонування програмно-технічних засобів виявлення вторгнень в IT-інфраструктури, виконати моделювання обробки підроблених пакетів при атаці на IT-інфраструктуру, побудувати модель атак на IT-інфраструктуру з вимірюваністю, побудувати модель атак на IT-інфраструктуру з керованістю;
5. розробити удосконалений метод виявлення вторгнень в IT-інфраструктури, що ґрунтується на застосуванні DES-моделей;
6. реалізувати програмно-технічні засоби виявлення вторгнень в IT-інфраструктури.

2 МОДЕЛЮВАННЯ ПРОЦЕСУ ВИЯВЛЕННЯ ВТОРГНЕНЬ В IT-ІНФРАСТРУКТУРИ

2.1 Дискретно-подійне моделювання вторгнень в IT-інфраструктури

З метою моделювання вторгнень в IT-інфраструктури в роботі було застосовано апарат дискретно-подійне моделювання (DES).

2.1.1 Пропонований активний IDS на основі DES для атак, пов'язаних з ARP

Розглянемо модель виявлення вторгнень на основі DES для атак, пов'язаних з ARP.

Зроблено наступні припущення.

1. Нескомпрометовані хости надсилатимуть одну відповідь на запит ARP протягом певного інтервалу T_{req}.
2. IDS працює на виділеній і надійній машині з фіксованою IP-адресою.
3. Дзеркальне відображення портів увімкнено на комутаторі, щоб IDS мав копію всіх вихідних і вхідних пакетів з усіх портів.

2.1.2Дослідження механізму активного зондування

Після отримання запитів ARP або відповідей ARP, ARP-зонди надсилаються на IP-адресу джерела запиту або пакету відповіді. Перед відправкою зондів перевіряється, що вихідна пара IP-MAC ще не перевірена детектором DES як справжня або підроблена. Деталі таких перевірених пар IP-MAC записані в таблицях, які розглядаються далі. Відтепер в обговоренні використовуються короткі позначення, як показано в Таблиці 2.1.

Таблиця 2.1 – Використані позначення

ShortNotation	Description
IPS	SourceIPAddress
IPD	DestinationIPAddress
MACS	SourceMACAddress
MACD	DestinationMACAddress
RQP	ARPrequestpacket
RSP	ARPresponsepacket
RQP_{IPS}	SourceIPofRQP;
PRQP	Proberequestpacket
PRSP	Proberesponsepacket

Подібно до RQP_{IPS}, підпис буде використовуватися для позначення IP-адреси призначення, MAC-адреси джерела та MAC-адреси призначення для RQP і RSP.

1. Кожного разу, коли запит ARP перевіряється на справжність, робиться запис у таблиці Authenticated, що позначається як AUTHT. AUTHT має п’ять полів: вихідний IP AUTHTIPS, вихідний MAC AUTHTMACS, IP AUTHTIPD призначення, MAC AUTHTMACD призначення та час, коли IDSAUTHTTiотримує пакет запиту.

2. Для підроблених запитів деталі вводяться в іншу таблицю, яка називається Spoofedtable (позначена як SPOOFT), яка має ті самі поля, що і таблиця Authenticated.

3. Кожного разу, коли відповідь ARP перевіряється як справжня (або підроблена), в таблицю Authenticated (Spoofed) робиться запис.

У разі запитів ARP можна заповнити лише чотири з п'яти полів у таблицях; MACD залишити як « − −», оскільки MAC-адреса призначення не пов’язана із запитами. Однак у разі відповідей усі п’ять полів заповнюються відповідними значеннями.

$(TableName)_{MAX}$представляє максимальну кількість елементів у таблиці в певний момент часу. Ці таблиці не тільки допомагають мінімізувати запити

ARP, але також використовуються для визначення того, чи спуфінг призвів до інших атак, таких як MiTM, DoS тощо.

Техніка зондування має два основних модулі, а саме: ARP REQUEST-HANDLER() і ARP RESPONSE-HANDLER().Алгоритм 2.1 обробляє пакети запитів ARP в мережі. Для будь-якого пакету запиту ARP RQP він спочатку перевіряє, чи є він неправильним (тобто будь-які зміни в незмінних полях заголовка пакета ARP або різних MAC-адрес у полях заголовків MAC і ARP) або одноадресним. Ці випадки є ненормальними, і рішення може бути прийняте за спостереженням за будь-яким таким (одиничним) пакетом, на відміну від послідовності пакетів ARP у разі спуфінгу.

Виявлення таких аномальних випадків є тривіальним, і не розглядаються при моделюванні DES та виявленні атак. Алгоритм 2.1 просто виходить (встановлюючи прапор статусу) для таких пакетів запитів ARP.

Якщо пакет не одноадресний або неправильно сформований, а пакет запиту від (IDS), тобто RQP_{IPS}- це IP IDS, а RQP_{MACS} - MAC IDS, Алгоритм 2.1 пропускає обробку цього пакета; не потрібно надсилати запити ARP на запити ARP від IDS, оскільки припускається, що IP-MAC-парування IDS відоме та підтверджене. Якщо жоден із наведених вище випадків не збігається, RQP_{IPS} виконується пошук у таблиці Authenticated. Якщо збіг знайдено як $AUTHT_{IPS}[i]$і відповідна вихідна MAC-адреса AUTHTMACS[i] у таблиці така ж, як і RQPMACS, пакет має справжню пару IP-MAC, яка вже визначена детектором. Ця справжність досягається без явного використання детектора DES. Крім того, в цьому випадку не надсилаються ARP-зонди, що економить деякий трафік ARP. RQP_{IPS},RQP_{MACS},RQP_{IPD},−− і Час отримання зберігаються в таблиці Authenticated. У разі збігу в IP-адресі джерела та невідповідності вихідної MAC-адреси (тобто, $RQP_{IPS}=AUTHT_{IPS}[i]$and$RQP_{MACS}-=AUTHT_{MACS}[i]$)виявляється, що пакет підроблений без явного використання детектора DES і не надсилаються ARP-зонди. Деталі RQP зберігаються в таблиці підроблених. З іншого боку, якщо RQPIPS не знайдено в AUTHT (у полі AUTHTIPS), то і RQPIPS, і RQPMACS шукають у SPOOFT (тобто RQP_{IPS} =$SPOOFT_{IPS}[i]$andRQP_{MACS}

=SPOOFT$_{MACS}$[i]). Якщо збіг знайдено, то це означає, що пара IP-MAC вже виявлена як підроблена. Проби ARP не надсилаються, а деталі пакету запиту додаються в SPOOFT. Якщо обидва наведені вище випадки не задовольняються, то Алгоритм 2.1 надсилає (широкомовний) запит на зонд ARP, який запитує MAC для вихідного IP RQPIPS. В іншому випадку, тобто при безоплатних запитах ARP, надсилається ARP-зонд. Безкоштовні запити ARP транслюються (наприклад, введення нового хоста або зміна карти NIC), щоб інформувати інші хости про його пару IP-MAC. Безоплатний запит ARP можна визначити, якщо RQP$_{IPS}$=RQP$_{IPD}$. Для такого безоплатного запиту ARP-зонд надсилається до RQPIPS без перевірки в таблицях, оскільки такі запити робляться, коли з'являється хост із «новою парою IP-MAC», який потрібно перевірити детектором заново.

У даному випадку розглядаються такі зміни як події: Отримання RQP, відправлення запиту на зонд ARP (PRQP), отримання відповіді на зонд ARP (PRSP), отримання відповіді (крім для ARP зонда) RSP. Ці запити/відповіді з відповідними полями (наприклад, IP джерела RSP) розглядаються як виміряні події моделі DES. Крім того, коли Algorithm2.1 вирішує, що деяка пара IP-MAC є справжньою/підробленою за допомогою аутентифікованої таблиці/підробленої таблиці, це означає те ж саме; це фіксується подією «виявлено (DTD)».

Алгоритм 2.2 – це обробник відповіді ARP. Для будь-якого пакету відповіді ARP RSP, алгоритмспочатку знаходить неправильно сформовані пакети і відповідно встановлює статус.

Алгоритм 2.1 – ОБРОБНИК ЗАПИТУ ARP

Input : RQP - ARP requestpacket
Output: Intimatereceiptof RQP, sendingof ARP proberequestanddetected DTD, Updated AUTHT and SPOOFT
if (RQP ismalformed) OR (RQP isUnicast) then
"Statusisabnormal" and EXIT

endif

if (RQPIPS = IP(IDS)) AND (RQPMACS = MAC(IDS)) then

EXIT

endif

if (RQPIPS =AUTHTIPS[i] AND RQPMACS = AUTHTMACS[i], forany i, 1 ≤ i ≤ AUTHTMAX)

then

“StatusisGenuine”;

add {RQPIPS, RQPMACS, RQPIPD, −−, timeofreceipt} in AUTHT;

Intimate RQP and DTD; EXIT

endif

if (RQPIPS =AUTHTIPS[i], forany i, 1 ≤ i ≤ AUTHTMAX) AND (RQPMACS! =

AUTHTMACS[i]) then

“StatusisSpoofed”;

add {RQPIPS, RQPMACS, RQPIPD, −−, timeofreceipt} in SPOOFT;

Intimate RQP and DTD; EXIT

endif

if (RQPIPS =SPOOFTIPS[i] and RQPMACS =SPOOFTMACS[i], forany i, 1 ≤ i ≤ SPOOFTMAX)

then

“StatusisSpoofed”;

add {RQPIPS, RQPMACS, RQPIPD, −−, timeofreceipt} in SPOOFT;

Intimate RQP and DTD; EXIT

else(/* Gratuitousi.e., RQPIPS == RQPIPD ornoneoftheaboveconditions */)

Intimatereceiptof RQP;

Send ARP ProbeRequest PRQP to RQPIPS from IDS; Intimate sendingofthe ARP proberequest PRQP;

endif

Далі перевіряється, чи є пакет-відповідь для будь-якого ARP-зонда (відправленого IDS). Відповідь на зонд ARP можна визначити, якщо RSPIPD є IP IDS, а RSPMACD є MAC IDS. Для цих пакетів не надсилається запит, і алгоритм повідомляє про отримання цих пакетів відповідей (PRSP).

Якщо відповідь не сформована і не відповідає на запит ARP, IDS надсилає запит на зонд ARP (широкомовну передачу) до вихідного IP RSPIPS, запитуючи його MAC. Перш ніж відправити зонд, Алгоритм 2.2 виконує перевірку в Аутентифікованій (Authenticated) таблиці та Підробленої (Spoofed) таблиці, подібну до тієї, що розглядається в Алгоритмі 2.1.

Алгоритм 2.2 – ОБРОБНИК ВІДПОВІДІ ARP()

Input:RSP-ARPresponsepacket
Output:IntimatereceiptofRSP,ARPproberesponsePRSP,sendingofARPproberequest
PRQPanddetectedDTD,UpdatedAUTHTandSPOOFT
if(RSP ismalformed) then
“Statusisabnormal” and EXIT
endif
if (RSPIPD = IP(IDS)) AND (RSPMACD = MAC(IDS)) then
Intimatereceiptof PRSP; EXIT
endif
if (RSPIPS =AUTHIPS[i] AND RSPMACS = AUTHMACS[i], forany i, $1 \leq i \leq$ AUTHTMAX)
then
“StatusisGenuine”;
add {RSPIPS, RSPMACS, RSPIPD, RSPMACD, timeofreceipt} in AUTHT;
Intimate RSP and DTD; EXIT
endif
if (RQPIPS =AUTHIPS[i], forany i, $1 \leq i \leq$ AUTHMAX) AND (RQPMACS! = AUTHMACS[i])
then

“StatusisSpoofed”;
add {RSPIPS, RSPMACS, RSPIPD, RSPMACD andtimeofreceipt} in SPOOFT;
Intimate RSP and DTD; EXIT
endif
if (RQPIPS =SPOOFTIPS[i] and RQPMACS =SPOOFTMACS[i], forany i, $1 \leq i \leq$ SPOOFTMAX)
then
“StatusisSpoofed”;
add{RSPIPS,RSPMACS,RSPIPD,RSPMACD,timeofreceipt}inSPOOFT;
IntimateRSPandDTD;EXIT
else(/*Gratuitousi.e.,RSPIPS==RSPIPDornoneoftheaboveconditions*/)Intimatereceiptof RSP;
Send ARP ProbeRequest PRQP to RSPIPS from IDS; Intimate sendingofthe ARP proberequest PRQP;
endif

2.1.3 Моделювання обробки підроблених пакетів

Розглянемо процес обробки підроблених пакетів відповіді за Алгоритмом 2.2. Крім того, цей приклад також підкреслює різницю в послідовності пакетів ARP (після активного зондування) у разі спуфінгу та звичайного senecio. Тут мережа має чотири хости A, B, D і E; E – це IDS, а D – зловмисник. Машинам A, B, D і E призначаються IP-адреси 10.0.1.1, 10.0.1.2, 10.0.1.4 і 10.0.1.5 відповідно. MAC-адреси хостів A, B, D і E: 08:4D:00:7D:C1, 08:4D:00:7D:C2, 08:4D:00:7D:C4 і 08:4D:00: 7D:C5 відповідно. Дзеркальне відображення портів увімкнено на комутаторі, щоб E мав копію всіх вихідних і вхідних пакетів з усіх портів. Крім того, E має мережевий інтерфейс виключно для надсилання запитів ARP та отримання відповідей на запити ARP.

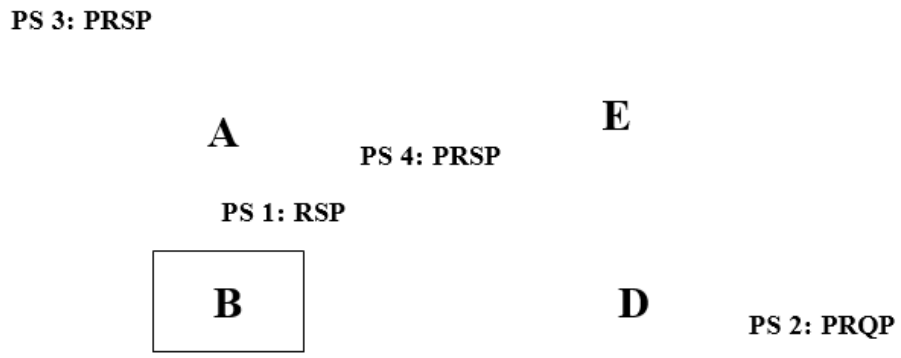


Рисунок 2.1 – Приклад підробленої відповіді

На рисунку 2.1 показано послідовність пакетів (позначена порядковим номером пакета), що вводяться, коли зловмисник D надсилає підроблену відповідь як «IP(B)-MAC(D)» хосту A та його перевірку (за допомогою активного зонда). Як пакетна послідовність 1, D надсилає відповідь ARP RSP до A, повідомляючи, що IP-адрес B пов'язаний з MAC-адресом D. Ця розмова відповіді ARP розглядається в tcpdump (одна із утиліт для перегляду пакетів, видимих для інтерфейсу) як:
08 :4D:00:7D:C4 08:4D:00:7D:C1 60: відповідь ARP 10.0.1.2 o 08:4D:00:7D:C4.

Оскільки ARP є протоколом без стану, після отримання відповіді від D, A оновлює свій кеш парою IP-MAC як IP(B)-MAC(D); кеш ARP A з оновленням показаний у Таблиці 2.2.

Таблиця 2.2 – ARP-кеш машини A під підробленою(Spoofed) відповіддю

IP	MAC	Interface
10.0.1.2	08:4D:00:7D:C4	eth0

Можна зазначити, що за звичайних умов «відповідь B на A з парою IP-MAC :IP B-MAC з B» і умова атаки «відповідь D на A з парою IP-MAC :IP B-MAC з D», Послідовність ARP-пакетів не змінюється. Під час атаки весь трафік, який A хоче надіслати до B, буде надіслано до D.

При отриманні RSP від D до A (пакетна послідовність 1), ARP RESPONSE HANDLER() в E, сповіщає про отримання RSP (подія 1), відправляє запит ARP PRQP на IP B для запиту відповідного MAC (послідовність пакетів 2).) і підказує надсилання запиту на зонд ARP PRQP (подія 2). Ця розмова із запитом на запит ARP розглядається в tcpdump як: 08:4D:00:7D:C5 ff:ff:ff:ff:ff:ff 42: ARP who-has 10.0.1.2 tell 10.0.1.5.

Запит на перевірку надіслано, оскільки вихідна пара IP-MAC цього RSP ще не перевірена (Автентифікована таблиця і Підроблена таблиця порожні). Оскільки зонд ARP є широкомовним, і B, і зловмисник D отримають зонд. B безумовно відповідатиме на зонд як PRSP з IP(B)-MAC(B) до E (пакетна послідовність 3), оскільки передбачається, що не входить хост (B) завжди відповідатиме на зонд. Ця розмова відповіді на запит ARP розглядається в tcpdump як:

08:4D:00:7D:C208:4D:00:7D:C560:ARPreply10.0.1.2is-at08:4D:00:7D:C2.

Тепер IDS може знати, що відповідь, зроблена в послідовності пакетів 1, є хибною (оскільки вона вже має IP(B)-MAC(D)), і IDS може генерувати тривогу (а також відстежувати MAC зловмисника (D)). Щоб уникнути самоідентифікації, зловмисник D повинен відповідати на всі запити щодо MAC IP(B) за допомогою IP(B)-MAC(D); ця відповідь на зонд ARP від D розглядається в tcpdump як:

08:4D:00:7D:C408:4D:00:7D:C560:ARPreply10.0.1.2is-at08:4D:00:7D:C4.

Можна відзначити, що порядок відповіді зловмисника і справжнього хоста не фіксований. Отже, якщо і зловмисник, і справжній хост відповідають на запит своїми MAC-ами, IDS може виявити підробку, але не може вказати MAC зловмисника. Отже, розумно вважати, що зловмисник відповість на всі запити щодо IP-адреси, яку він фальсифікує. Підсумовуючи, принаймні одна відповідь на запит (скажімо, надсилання для перевірки IP(B), який очікується надійде і матиме справжній IP-MAC B. У разі підробки очікується надходження більше відповідей, які можуть мати різні MAC. Після виявлення спуфінгу в таблицю підробок робиться запис, як показано в таблиці 2.3.

У цьому прикладі було припущено, що справжній хост B відповідає (послідовність пакетів 3) перед зловмисником D (послідовність пакетів 4) на зонд ARP. Відповіді в послідовності пакетів 3 і пакетної послідовності 4 обробляються ARP RESPONSE HANDLER() як «близьке отримання PRSP», таким чином генеруючи подію 3 і подію 4 відповідно. Можна зазначити, що в двох відповідях різні MAC-адреси пов'язані з IP-адресою, яка досліджується, тобто коли MAC(B) пов'язано з IP(B), а потім MAC(D) пов'язано з IP(B). У таблиці 2.4 наведено послідовність пакетів ARP для прикладу (на рисунку 2.1).

Таблиця 2.4 – Таблиця послідовності пакетів і подій у прикладі

PS:Events	SRCIP	SRCMAC	DestIP	DestMAC
PS1:RSP	10.0.1.2	08:4D:00:7D:C4	10.0.1.1	08:4D:00:7D:C1
PS2:PRQP	10.0.1.5	08:4D:00:7D:C5	10.0.1.2	–
PS3:PRSP	10.0.1.2	08:4D:00:7D:C2	10.0.1.5	08:4D:00:7D:C5
PS4:PRSP	10.0.1.2	08:4D:00:7D:C4	10.0.1.5	08:4D:00:7D:C5

Розглянемо послідовність ARP-трафіку, якщо відповідь у послідовності пакетів 1 є справжньою, тобто IP(B)-MAC(B) надсилається до A від B. Для ARP-зонда, надісланого для перевірки IP(B)-MAC (B) (пакетна послідовність 2) тільки B відповість на E IP(B)-MAC(B) (пакетна послідовність 3). До часу Treq IDS отримає лише один PRSP (на відміну від двох у разі атаки). Як тільки детектор DES (тобто IDS) виявляє, що відповідь є справжньою (за допомогою лише одного PRSP), деталі RSP зберігаються в таблиці Authenticated. Якщо в рамках Treq отримано більше одного PRSP з різними MAC, деталі RSP зберігаються в таблиці підроблених. Після цього запити не надсилатимуються для відповіді/запиту ARP із IP-адресою джерела як IP(B). Однак у разі безоплатного запиту/відповіді буде надіслано запит ARP.

Як обговорювалося раніше, механізм IDS для виявлення атак підробки ARP може бути створений за допомогою детектора DES. Отже, спочатку події ARP у звичайних і підроблених умовах моделюються за допомогою моделей DES на основі подій, а потім розробляється детектор DES.

Перш ніж розглянути формальну структуру моделювання DES, необхідно проілюструвіти абстрактні моделі на основі подій для ARP у звичайних і підроблених умовах. На рисунку 2.2 показана модель ARP на основі стану в нормальних умовах. Перераховані раніше події, а саме отримання RQP/RSP, відправлення запиту на зонд ARP (PRQP) та отримання відповіді на зонд ARP (PRSP), показані жирним шрифтом із переходами на малюнку. Далі, деяка розробка подій також проілюстрована переходами. Рисунок 2.3(A) і рисунок 2.3(B) показують модель відповідно до підміни запитів і підробок відповідей.

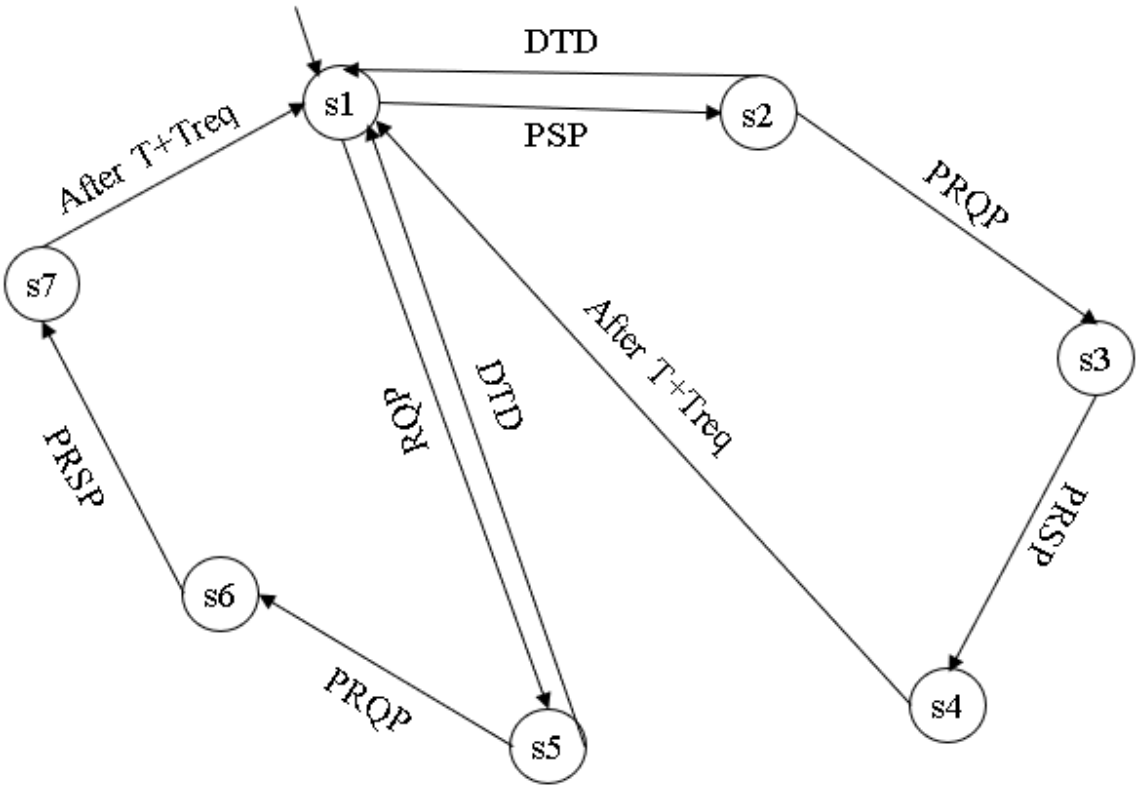
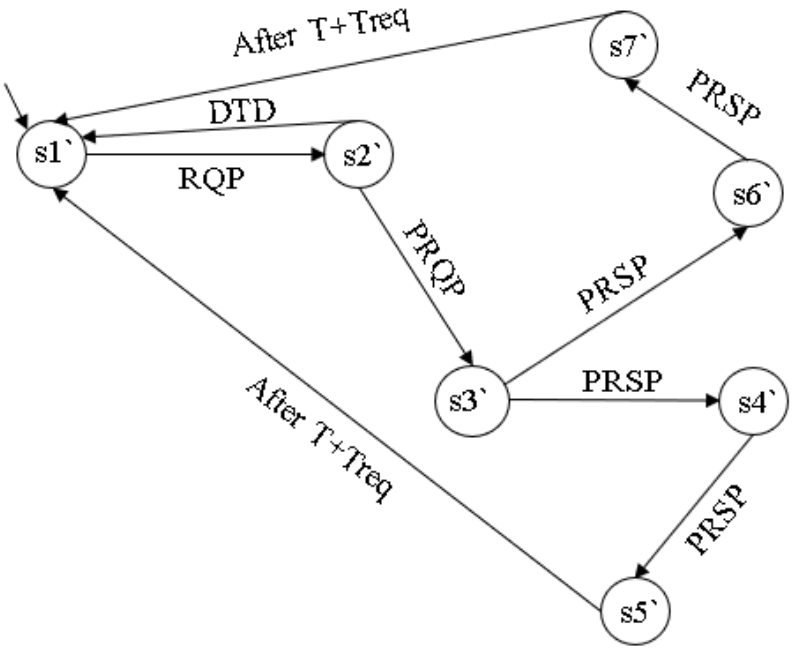


Рисунок 2.2 – Модель ARP на основі стану в нормальних умовах

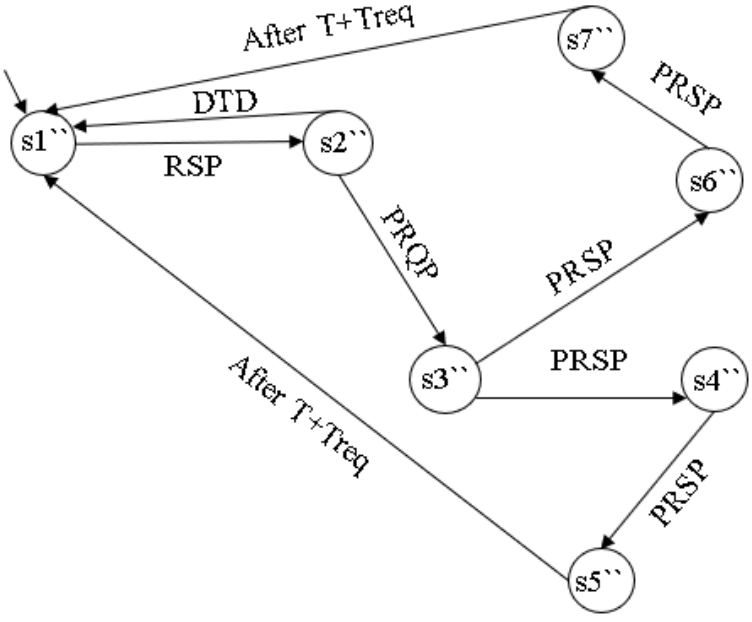
2.4 Активне моделювання DES

Розглянемо використання активного фреймворка DES для виявлення атак підробки ARP. Структура FDD для активного DES, запропонована в [21], була адаптована для виявлення ARP-атаки. Розширення теорії, представленої в [21], було зроблено для застосування її при розробці IDS для ARP-атак. Далі перераховані мотиви та розширення.

1. Розмір доменів змінних, які беруть участь у моделюванні DES для IDS, дуже великий порівняно із системами, які зазвичай обробляються структурами DES, такими як системи насос-клапан, хімічні заводи тощо (порядки тисяч). З іншого боку, у випадку IDS, наприклад, розмір домену IP-адреси може бути O(256.256.256.256). Таким чином, в адаптовану активну модель DES, крім змінних стану, також включені змінні моделі.



а) спуфінг запит



б) спуфінг відповідь

Рисунок 2.3 – Модель ARP на основі стану під підробкою запитів і підробкою відповідей

2.2.1 Адаптований активний DES моделювання

Активна модель DES G являє собою шість кортежів (Σ, X, X0, V, C, f5), де Σ – множина подій, S – множина станів, X0 ⊆ X – множина початкових станів, V – набір змінних моделі, C – набір змінних тактового сигналу, а f5 – набір переходів. Можна відзначити, що остаточного стану немає, оскільки трафік ARP є процесом напівоновлення (ніколи не зупиняється, поки мережа працює). Існують деякі стани, з яких існує перехід до початкового(их) стану(ів), що представляє оновлення процесу; такі стани називаються станами оновлення. Отже, модель DES працює.

Кожен елемент vi з V (V = {v1, v2, · · · , vn}) може приймати значення з області Di. Змінні годинника приймають значення в невід'ємних дійсних числах R. Інваріантна умова для тактової змінної c, позначена як Φ(c), є лінійною нерівністю або рівністю змінної з невід'ємним дійсним.

Перехід T ∈ f5 – це сім кортежів $(x,x^1,\sigma,\varphi$ (V), Φ(C), Reset(C), Assign(V)), де x – вихідний стан, x^1– стан призначення, σ– подія (для якої запускається перехід), φ(V) – булева кон'юкціярівностей підмножини змінних у V, Φ(C) – інваріантна умова на підмножину тактових змінних, Reset(C) – це підмножина змінних годинника, які потрібно скинути, а Assign(V) – це підмножина змінних моделі та присвоєнь зі значеннями з відповідних доменів.

Слідом моделі DES G є послідовність переходів, породжених G і позначених як $\Delta s = (\tau_1,\tau_2,)$, де initial($\tau_1$) є початковим станом у X0 і виконується властивість послідовності, тобто , initial(τ_1+1) = final(τ_1), для i ≥ 1. Сліди нескінченні, і кінцевий префікс сліду називається «кінечний слід». Відтепер приймемо властивість послідовності для будь-якої «послідовності переходів». Для будь-якого сліду s = (τ_1, τ_2, · · ·), initial(s) = initial(τ_1) і для кінцевого префікса s = (τ_1, τ_2, · · · τ_f), final(τ_f) = final(s). Говорять, що стан x знаходиться в сліді s, якщо x=initial(τ_i),для деякого i ≥ 1. Множина всіх слідів,

породжених G, та їх скінченних префіксів є мовою G, позначеною як L(G) . Множина Lf (G) позначає підмножину L(G), що містить скінченні префікси членів L(G). Природно, L(G) − Lf (G) – це підмножина $f5^{W}$, де $f5^{W}$– множина всієї нескінченної послідовності f5; Lf (G) є підмножиною $f5^{*}$, замиканням Kleene f5.

Постова мова G після кінцевого префікса s сліду, позначеного як L(G)/s, визначається як $L(G)/s = \{t \in f5^{W} \cup f5^{*} | st \in L(G)\}$. $Lf(G)/s \subset L(G)/s$ містить скінченні префікси слідів L(G)/s.

Переходи поділяються на вимірювані (позначаються як $f5_{u}$) і невимірні підмножини (позначаються як $f5_{um}$). Також транзакції класифікуються як контрольовані (позначаються як $f5_{c}$) і неконтрольовані (позначаються як $f5_{uc}$) підмножини.

2.2.2 Модель з вимірюваністю

Щоб охопити обмеження вимірювань, набір переходів розбивається на дві непересікаючі підмножини, $f5_{u}$і $f5_{um}$, вимірних і невимірних відповідно. Прикладом вимірюваних переходів може бути відповідність подіям, таким як отримання пакетів запиту ARP, відправка пакетів із зондуванням тощо, тоді як прикладом невимірного переходу може бути запуск атаки (тобто подія, що спричиняє збій). Можна відзначити, що запуск атаки не може спостерігатися безпосередньо і його необхідно завершити послідовністю інших спостережуваних переходів. У невимірному переході ($\tau_{um} \in f5_{um}$) визначаються лише вихідний стан, стан призначення та ініційна подія, тобто τ_{um}:$(x,x^{1},\sigma,-,-,-,-)$. Таким чином, увімкнення τum залежить лише від σ, а не від будь-яких змінних моделі чи умови інваріантності годинника; це зображено символом « − » на переході. Крім того, τum не призначає змінні моделі та не скидає змінні годинника.

Приймемо два переходи $(x1,x^{1},\sigma1,\varphi1(V),\Phi1(C),Reset1(C),Assign1(V))i\tau2=(x2,x^{1},\sigma2,\varphi2(V),\Phi2(C),Reset2(C$

),Assign2(V)) еквівалентні, якщо ifσ1=σ2(та сама подія), φ1(V)≡φ2(V)(однакові рівності щодо ідентичної підмножини змінних у V), Φ1(C)≡Φ2(C) (точна інваріантна умова для змінних годинника), Reset1(C) ≡ Reset2(C) (ідентична підмножина змінних годинника скидається) та Assign1(V)≡Assign2(V)(та сама підмножина змінних моделі з ідентичним призначенням). Якщо τ1≡τ2, то вихідні стани переходів еквівалентні, а також стани призначення, тобто x1≡x2 і $x^1 \equiv x^1$.

Стан джерела та стани призначення невимірного переходу є еквівалентом вимірювання; оскільки виникнення невимірного переходу не може бути ідентифіковано, так і зміна стану також не може бути виявлена.

Приймемо оператор проекції P : $f5^* \rightarrow f5^*$можна визначити таким чином: P(ε)=ε;P(τ)=τ,τ∈$f5_m$;P(τ)=ε,τ∈$f5_{um}$;P(sτ)=P(s)P(τ),s∈$L_f(G)$,τ∈f5,де € – нульовий рядок.

Оператор P стирає невимірні переходи з траси кінцевого аргументу. Термін P(s) називається вимірним кінцевим слідом, що відповідає кінцевому сліду s.

Приймемо два кінцевих сліди (або послідовність переходів) sis^1 є еквівалентними вимірювання, якщо P(s)=(τ1,τ2,...τn),P(s^1)=(τ^1,τ^1,...τn) і $\tau_i E \tau^1$, 1≤i≤n.

Приймемо E для позначення еквівалентності вимірювань скінченних слідів, а також переходів, з невеликим зловживанням позначень. Оператор зворотної проекції P−1 : f5∗→ 2^{f5} визначається як $P^{-1}(s)=\{s^1 \in L_f(G) | sEs^1\}$Таким чином, $P^{-1}(s)$ включає всю можливу послідовність переходів, еквівалентну кінцевий слід s. Оператор проекції P, оператор зворотної проекції P^{-1} і поняття еквівалентності вимірювання E скінченних слідів можна природним чином поширити на сліди $\in f5^w$.

2.2.3 Модель з керованістю

Приймемо Перехід $\tau_c \in f5_c$ називається керованим, якщо відповідна подія σ може керуватися системним контролером.

При некерованому переході $\tau_{uc} \in f5_{uc}$ відповідна подія σ не може керуватися системним контролером.

Для стану x∈S з одним переходом τ, що виходить з нього, τ може бути керованим або некерованим, залежно від контексту. Однак, якщо врахувати стан x∈S, з якого, скажімо, виходить більше одного переходу $\tau 1, \tau 2, \cdots, \tau n$, якщо хоча б один перехід τ_i, $1 \leq i \leq n$ є некерованим, так і всі інші переходи, що виходять від x. Нехай $\tau 1$– некерований, а $\tau 2$– керований. Оскільки тригер $\tau 1$ не можна вимкнути, він може спрацювати до $\tau 2$, що робить його також некерованим.

За умови кінцевого префіксного сліду $s = (\tau 1, \tau 2, \cdots, \tau n)$ моделі G, s можна виключити з L(G), тобто $L(G) = L(G) - s$, якщо хоча б один перехід у $s = \tau 1, \tau 2, \cdots, \tau n$ є керованим. Якщо будь-який перехід τi в трасі s, що походить від x, є керованим, то подія, що відповідає τi, може бути вимкнена, коли система знаходиться в стані x, таким чином усуваючи s з L(G). Так само виконується для нескінченних слідів.

2.2.4 Моделювання відмов

Кожному стану x присвоюється мітка збою невимірною змінною статусу C з її доменом $= = \{N\} \cup 2^{\{F1, F2, \cdots, Fp\}}$, де $Fi, 1 \leq i \leq p$, означає постійний статус відмови та N означає нормальний стан.

Приймемо G-стан x нормальним, якщо x(C)= {N}. Множина всіх нормальних станів позначається як X_N.

Приймемо G-стан x станом відмови, або синонімом F_i -станом, якщо $F_i \in$ x(C). Множина всіх Fi-станів позначається як X_{Fi}.

Приймемо G-перехід (x, x^+) нормальним G-перехідом, якщо $x, x^+ \in X_N$.

Приймемо G-перехід (x,x^+) F_i -G-перехід, якщо $x,x^+ \in X_{Fi}$.Перехід (x,x^+), де $x(C)-=x^+(C)$, називається переходом з невдачею, що вказує перша поява деякої відмови у множині $x^+(C)-x(C)$. Оскільки збої вважаються постійними, немає переходу з будь-якого стану в xFi до будь-якого стану в x_N або від будь-якого стану в x_{FiFj} до будь-якого стану в x_F. Також для переходу (x,x^+) Зривні переходи непомітні і неконтрольовані.

Нехай $\Psi(X_{Fi}) = \{ s \in L_f(G) |$ останній перехід s є вимірним і $final(s) \in X_{Fi}\}$.

Кажуть, що модель G DES може діагностуватися за допомогою F_i на відмову F_i при обмеженні вимірювання, якщо виконується наступне

$\exists n \in N$ c.т. $[\forall s \in \Psi(XFi)\{\forall t \in Lf(G)/s(|t| \geq n \Rightarrow D)\}]$,

де умовою D є $\forall u \in P^{-1}[P(st)]$, $final(u) \in X_F$.

Наведене вище визначення означає наступне. Нехай s – будь-який скінченний префікс сліду G, що закінчується у F_i -стані, і нехай t – будь-яке досить довге продовження s. Тоді умова D вимагає, щоб кожна послідовність переходів, вимірювання еквівалентна st (тобто, що належить до $P^{-1}(P(st)))$,закінчувалася в F_i -стані. Це означає, що вздовж кожного продовження t з s можна виявити виникнення відмови, що відповідає F_i, протягом кінцевої затримки, або, точніше, протягом щонайбільше n переходів після s.

Нехай є два сліди $s_1 i s_2$, які порушують F_i-визначення діагностичності;$P^{-1}(P(s_1)) = P^{-1}(P(s_2))$ and $P^{-1}(P(s_1))$закінчується в F_i -стані, тоді як $P^{-1}(P(s_2))$закінчується в не -F_i-стан. Якщо $s_1 i s_2$ є єдиними слідами, які порушують діагностику, і s_2 буде усунено, тоді модель стане діагностичною.

Для моделі DES G з керованими та некерованими переходами усунути мінімальну кількість слідів S з L(G) так, що L(G) − S можна діагностувати за визначенням 2.9.

2.3 Перевірка моделі

Здійснимо перевірку (0діагностику моделі). Діагностика представлена у вигляді орієнтованого графіка O = (Z, A), де Z – набір вузлів діагноста, які називаються O-вузлами, а A – набір переходів діагностика, які називаються O-переходами. Кожен O-вузол z ∈ Z являє собою набір G-станів, що представляють невизначеність щодо фактичного стану, а кожен O-перехід a ∈ A виду (z_i,z_f) є набір еквівалентних переходів, що представляють невизначеність щодо фактичного вимірного перехід, що відбувається.

Приймемо невимірний наступник і невимірне охоплення набору G-станів):

Невимірний наступник (множина) множини Y станів визначається як U (Y) = {x∈Y{ x|τ= (x,x^+) ∈$f5_u$ }. Невимірне охоплення множини Y G-станів, позначене як U^*(Y), є рефлексивно-транзитивним замиканням невимірних наступників Y.

Діагностика будується, починаючи з початкового стану(ів) моделі. Стани в X_0розбиті на еквівалентні підмножини, позначені як X01, X02, · · · X0m. Для всіх i, 1 ≤ i ≤ m, початковий O-вузол z0i виходить як невимірне охоплення X0i, тобто z0i = U^*(X0i). Множину всіх початкових O-вузлів позначимо як Z0 = z01, · · · , z0m. Початкові O-вузли фіксують той факт, що діагностик може зробити висновок про набір z0i можливих початкових станів системи (або їх невимірного охоплення), вимірюючи змінні, не чекаючи першого вимірюваного переходу. З огляду на будь-який O-вузол z, O-переходи, що виходять з z, отримуються таким чином. Позначимо $f5_{mz}$ множину вимірних G-переходів із станів x ∈ z. Нехай A_Z– множина всіх класів еквівалентності $f5_{mz}$ щодо E. Для кожного a ∈A_Z можна створити наступний O-вузол z^+ з z, для якого z^+ = final(a) можна створити наступним чином. Нехай z^+ = { final(τ)| τ∈ a}; тоді z^+ = U^* (z^+) і a позначається як: (z,z^+). Набір переходів діагностики збільшується як: A ← A ∪ {a}, а набір O-вузлів – як: Z ← Z ∪ {z+}. Кожне a ∈ A є впорядкованою

парою (z,z^+), де z = initial(a) і z^+ = final(a). Таким чином, кожен O-вузол містить еквівалентні стани.

Відтепер стани, переходи та сліди G називатимемо відповідно G-станами, G-переходами та G-слідами. Аналогічно, для вузлів і переходів використовуються «O-вузли» та «O-переходи» відповідно.

Приймемо O-вузол, який містить F_i-стан, F_i-O-вузлом, позначається як zF_i.

ПриймемоF_i-O-вузол z F_i-певним O-вузлом, якщо $z \subseteq X_{F_i}$. F_i-O-вузол, який не є F_iі-певним, називається F_i-невизначеним.

Шлях діагноста O є послідовністю O-переходів γ = (a1, a2, ...), з властивістю послідовності. Можна відзначити, що відповідному будь-якому O-шляху γ = (a1, a2, · · ·), існує унікальна послідовність O-вузлів (z1, z2, · · ·), де z_i = initial(c) і z_{i+1}= final(a_i), для i ≥ 1. Отже, термін «O-шлях» використовується як взаємозамінні для послідовності O-переходів і послідовності O-вузлів. Аналогічно, «G-сліди» використовуються як взаємозамінні для послідовності G-переходів і послідовності G-станів.

Приймемо невизначений цикл – це Fi-O-цикл, у якому немає F_i-певного O-вузла.

Приймемо F_i-невизначений цикл γ, у якому F_i стани, що містяться в O-вузлах γ, утворюють цикл у G, що містить переходи від γ, називається F_i-невизначеним циклом.

Еквівалентність між F_i-діагностичною та відсутністю F_i-невизначених O-циклів була офіційно встановлена для моделей DES [20]. F_i-невизначені O-вузли містять деякі нормальні G-стани і деякі F_i-G-стани. Таким чином, невизначені O-вузли F_i не можуть визначити, чи працює система нормально, чи сталася несправність. З іншого боку, F_i-певні O-вузли можуть виявити, що сталася несправність, оскільки Fi-певні вузли містять лише F_i-G-стани. Наявність F_i-невизначених циклів означає, що після відмови F_i система може невизначено переміщатися в F_i-невизначених O-вузлах, що робить несправність не діагностованою. При активній діагностиці мінімальна кількість слідів від

L(G) повинна бути усунена (якщо можливо, через керованість), щоб не було F_i-невизначених циклів. Іншими словами, у будь-якому F_i-невизначеному циклі γ послідовність (слід) стану F_i, що міститься в O-вузлах γ, які утворюють цикл в G, що містить переходи від γ, має бути усунена (якщо можливо).

Тепер наведемо абстрактний приклад для ілюстрації проблеми активної діагностики. Рисунок 2.5(a) ілюструє активну модель DES абстрактної системи. Модель має сім станів, а саме a,b,c,a^1,b^1,c^1,d^1 де всі незаправлені (заправлені) стани представляють нормальну (відмовну) поведінку. Іншими словами, $C(a)=C(b)=C(c)=NiC(a^1)=C(b^1)=C(c^1)=C(d^1)=F_i$. Пунктирний перехід (збій) показує виникнення відмови Fi.

Припустимо, що переходи $1E1^1,2E2^1$і $3E3^1$, таким чином, стани aEa^1,bEb^1icEc^1.Невдача – це єдиний непомітний перехід. Помилки переходів, $1,2,1^1,2^1$вважаються некерованими, а $3,3^1,4^1$– керованими переходами.

Діагностика для моделі DES на рисунку 2.5(a) показана на рисунку 2.5(b). Нижче наведено деякі початкові кроки побудови діагноста для цього прикладу.

(i) Початковий стан діагностика, тобто z1, отримують наступним чином. X0 розбивається на еквівалентні вимірювання підмножини G-станів, які в цьому випадку один, тобто X0 = a; існує лише один початковий G-стан a (як показано на малюнку 2.5(a)) і X01 = {a}. Оскільки X0 можна розділити лише на одну підмножину еквівалентних вимірювань початкових G-станів, $z01=U^*(\{S01\})=U^*(\{a\})=\{a,a^1\}$.Таким чином, існує лише один початковий O-вузол z1 (як показано на рисунку 2.5(б)) і $z_1 = \{ a,a^1 \}$.

(ii) Вихідні O-переходи від z1 отримують наступним чином. Тут $f5_{mz1}=\{1,1^1\}$ що – усі вихідні вимірювані переходи з G-станів у z1. Тепер $A_{z1}=\{\{1,1^1\}\}$як $1E1^1$. Відповідно до $\{1,1^1\}$ є O-перехід a1.

O-вузол призначення, відповідний a1, отримують наступним чином. $z^+a1=\{b,b^1\}$, оскільки a1 містить G-переходи $1,1^1$ і final(1) = b і final(1^1)=b^1. Далі, $Z_1^+ = \{b, b^1\}$як $U^*(\{b\}) =\{b,b^1\}$ та $U^*(\{b^1\}) =\{b^1\}$.Таким чином, O-

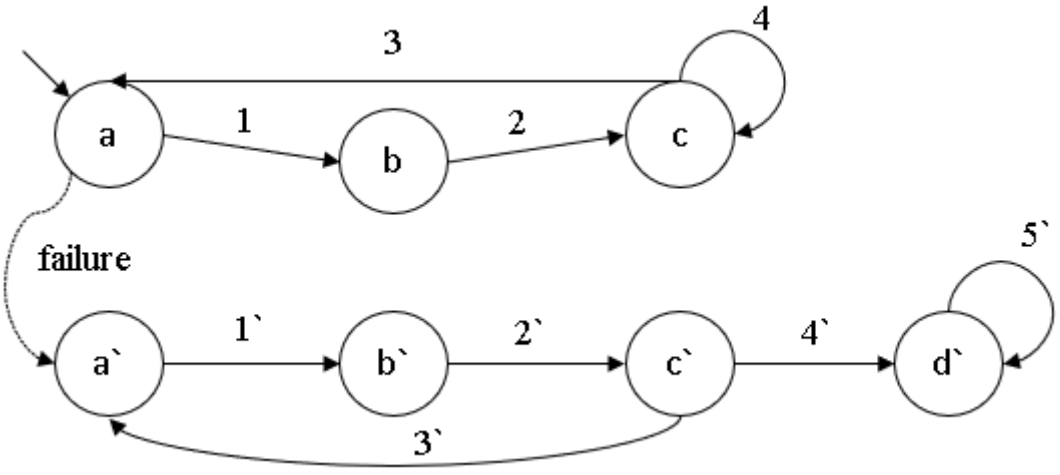
вузлом призначення O-переходу a1 є z_2 : $\{b, b^1\}$. Аналогічно, побудова діагностика продовжується до тих пір, поки на етапі (iii) не буде створено нових O-вузлів.

Можна відзначити, що існує F_i-невизначений цикл (z_1,z_2,z_3) = γ, скажімо (рисунок 2.5(b)), що робить несправність недіагностованою.

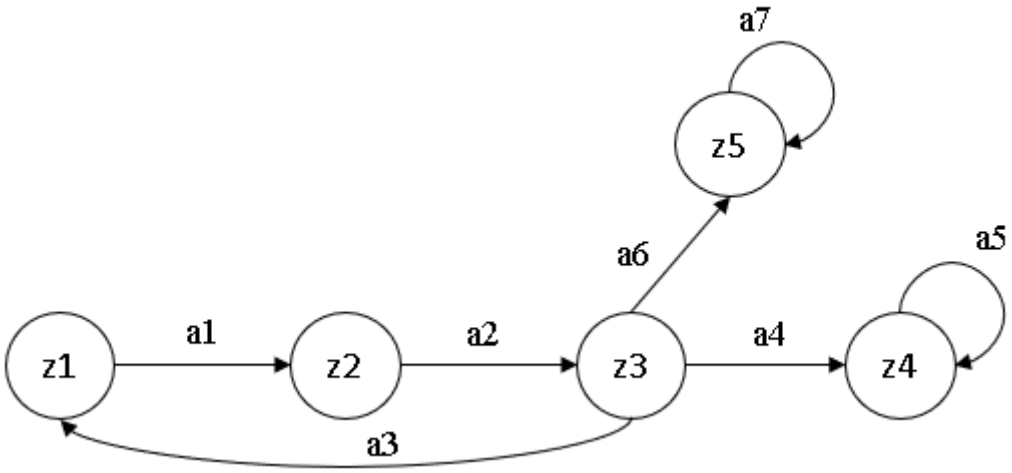
Теорію активної діагностики можна застосувати, щоб побачити, чи можна діагностувати збій, усунувши деякі сліди.

На рисунку 2.5(b) послідовність F_i-станів (слід), що міститься в O-вузлах γ (тобто z_1,z_2,z_3), які утворюють цикл в G, що містить переходи від γ (тобто a1,a2,a3),це(1^1,2^1,3^1). Оскільки 3^1 є керованим переходом, його початкова подія може бути вимкнена, коли система знаходиться в стані c^1(абосяксЕc^1), що виключає ((11, 21, 31)∗) з мови моделі; модель з вилученим слідом показана на рисунку 2.5(в), а відповідний діагностик на рисунку 2.5(г).

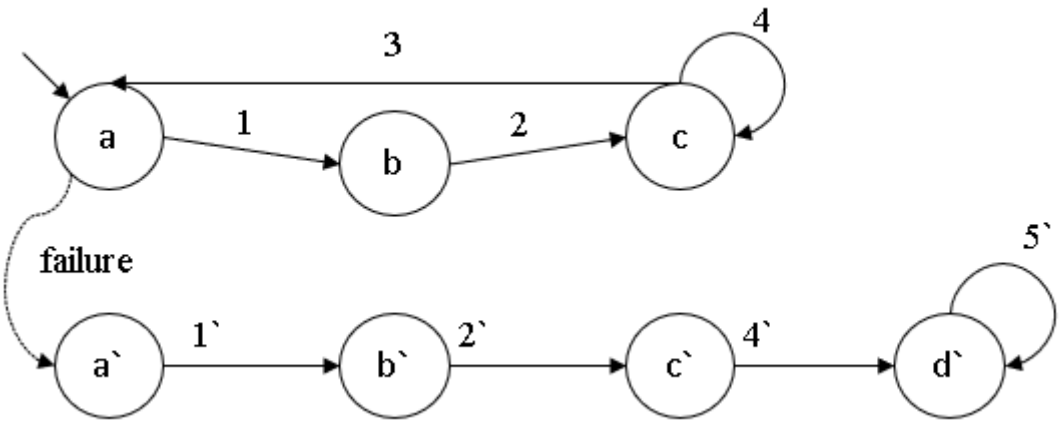
На рисунку 2.5(г) видно, що не існує циклу F_i-невизначеного, що дозволяє діагностувати F_i. Іншими словами, щоб зробити F_iдіагностованим, коли система знаходиться в стані c^1(абосяксЕc^1), контролер повинен увімкнути подію для переходу 4^1 і подію вимкнення для переходу $4^{\,1}$.



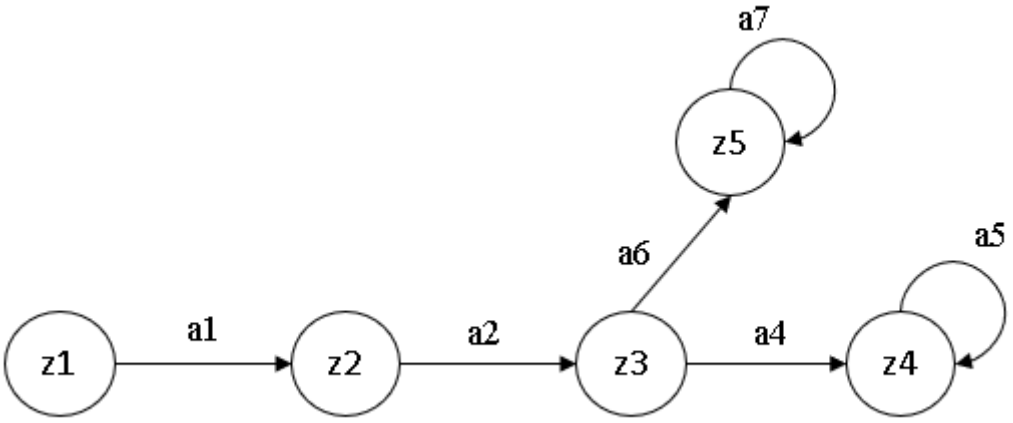
а) активна DES модель



б) діагностика для DESмоделі



в) активна DESмодель після трасувння



г) діагностика для DESмоделі після трасувння

Рисунок 2.5 – Активна модель DES для діагностики ARP

2.4 DES-моделювання атаки типу спуфінгу ARP

Рисунок 2.6 ілюструє активну модель DES для ARP із зондуванням у звичайному режимі та підробкою запитів.

Значення різних параметрів в активній структурі DES для моделювання ARP є наступним.

Σ = {RQP, RSP, PRQP, PRSP, attack}. Сукупність станів S показано на рисунку2.6. Стани без простих чисел відповідають нормальній ситуації, а стани з одним простим числом позначають підробку запиту. Початковий стан X0 = x_1. Набір змінних моделі V = {IPS, IPD}, і обидва його елементи мають ту саму область, задану як $D1(=D2)=\{d.d.d.d|d\in\{1,2,\cdots,255\}\}$. Існує одна змінна годинника y. Переходи показані на рисунку 2.6; як і стани, переходи без простих чисел призначені для звичайної моделі, а одиночне просте – для підробки запитів. Можна зазначити, що для деяких полів у кортежі, що представляють переходи, є «-». Якщо “ − ” для φ(V) або Φ(C), то представляє умову TRUE, тоді як якщо “–” для Reset(C) або Assign(V), то представляє NO дію (тобто, скидання або призначення) слід взяти. Керовані переходи позначені як u, а некеровані – uc. Збій, що спричиняє перехід $x1,x1$,attack,–,–,–,–(тобто, що спричиняє підробку запиту), є єдиним переходом, який не можна виміряти. Огляд активної моделі DES для ARP у звичайних випадках і випадках підробки запитів наведено нижче.

Звичайний випадок. У нормальному випадку (Рисунок 2.6) модель переходить з x1 в стан x2 при спостереженні події RQP шляхом переходу $\tau 1$; цей перехід неконтрольований, оскільки будь-який справжній хост або зловмисник може надіслати пакет відповіді іншому хосту. Увімкнення залежить лише від RQP, а не від будь-яких змінних моделі чи умови інваріантності годинника; це зображено символом « − » на переході.

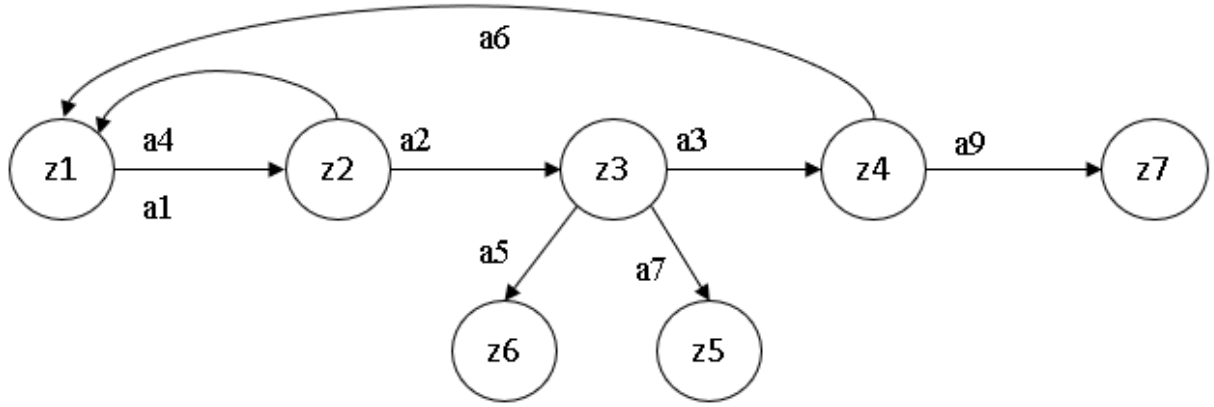


Рисунок 2.6 – Активна модель спуфінгу ARP на основі DES у звичайних умовах та в умовах атаки

Змінним моделі IPS і MACS присвоюються IP-адреса джерела та MAC-адреса джерела RQP відповідно. Змінна годинника y скидається на 0. Після цього в стані x_2є два варіанти: або надсилається датчик ARP (PRQP) (τ2), або система повертається до стану x1 після часу Treq (τ5) отримання RQP . Перехід τ2 увімкнено на PRQP (відправляється на IP-адресу джерела запитуваного RQP); Відповідність PRQP з RQP визначається перевіркою змінної моделі IPS з PRQPIPD. Також змінна годинника y скидається. Можна зазначити, що рішення про надсилання PRQP приймається контролером на основі таких умов, як (i) PRQP надсилається, якщо вихідний IP-MAC RSP є новим (ще не перевірено), (ii) PRQP не надсилається, якщо додатковий трафік в мережі через зондування необхідно мінімізувати тощо. Отже, τ2 є керованим переходом. Якщо PRQP не надіслано, система чекає в стані x2 для Treq, а потім переходить до x1 через τ5; Інваріантна умова годинника y >Treq в τ5 визначає, що час Treq пройшов. Оскільки поява τ5 залежить від τ2 (який є керованим), τ5 також є контрольованим. Після переходу τ2 надходить відповідь зонда (PRSP) від (нормального) хоста (τ3); відповідність PSQP з RQP забезпечується перевіркою IPS = PRSPIPS і MACS = PRSPMACS. Крім того, PRSP має надійти протягом часу Treq після надсилання PRQP; це перевіряється умовою інваріантності

годинника y ≤ Treq у переході τ3. PRSP надсилаються хостами і є неконтрольованими з точки зору IDS, що робить перехід τ3 некерованим. У звичайній ситуації на запит на зонд надходить лише одна відповідь на зонд. Отже, після отримання відповіді на один зонд модель повертається до вихідного стану x1 шляхом переходу τ4. τ4 спрацьовує миттєво після того, як система досягає x4, тому немає жодної активної події чи умови. Через відсутність умови дозволу τ4 є некерованим переходом.

Змінним моделі IPS і MACS призначаються IP-адреса джерела та MAC-адреса джерела RQP відповідно. Змінна годинника y скидається на 0. Після цього в стані x_2 є два варіанти: або надсилається датчик ARP (PRQP) (τ2), або система повертається до стану x_1 після часу T_{req}(τ5) отримання RQP.

Перехід τ2 увімкнено на PRQP (відправляється на IP-адресу джерела запитуваного RQP); Відповідність PRQP з RQP визначається перевіркою змінної моделі IPS з $PRQP_{IPD}$. Також змінна годинника y скидається.

Можна зазначити, що рішення про надсилання PRQP приймається контролером на основі таких умов, як (i) PRQP надсилається, якщо вихідний IP-MAC RSP є новим (ще не перевірено), (ii) PRQP не надсилається, якщо додатковий трафік в мережі через зондування необхідно мінімізувати тощо. Отже, τ2 є керованим переходом.

Якщо PRQP не надіслано, система чекає в стані x_2 для T_{req}, а потім переходить до x_1 через τ5; Інваріантна умова годинника y > T_{req} в τ5 визначає, що час Treq пройшов. Оскільки поява τ5 залежить від τ2 (який є керованим), τ5 також є керованим. Після переходу τ2 надходить відповідь зонда (PRSP) від (нормального) хоста (τ3); відповідність PSQP з RQP забезпечується перевіркою IPS = $PRSP_{IPS}$ і MACS = $PRSP_{MACS}$ Крім того, PRSP має надійти протягом часу T_{req} після надсилання PRQP; це перевіряється умовою інваріантності годинника y ≤ T_{req} у переході τ3. PRSP надсилаються хостами і є неконтрольованими з точки зору IDS, що робить перехід τ3 некерованим. У звичайній ситуації на запит на зонд надходить лише одна відповідь на зонд. Таким чином, після отримання відповіді на один зонд модель повертається до

вихідного стану x_1шляхом переходу τ4. τ4спрацьовує миттєво після того, як система досягає x_4, тому немає жодної активної події чи умови. Через відсутність умови дозволу τ4 є некерованим переходом.

Випадок підробки запиту. Заправлені стани та переходи на рисунку 2.6 представляють підробку запитів ARP із дослідженням IDS. Модель переходить у стан x_2^1 при спостереженні події RQP переходом T_1^1. Змінним моделі IPS і MACS призначаються IP-адреса джерела та MAC-адреса джерела RQP відповідно.

Змінна годинника y скидається на 0. Як і T_1., T_1^1. є некерованим переходом. Після цього в стані x_2^1 є два варіанти: або надсилається зонд ARP (PRQP) (T_2^1), або система повертається до стану x_1 після часу T_{req} (T_1^1.) отримання RQP. Як і у випадку нормального стану, T_2^1 і T_7^1 є керованими переходами. Після переходу T_2^1 є три варіанти – (i) приходить відповідь на зонд від зловмисника (T_5^1), що має $PRSP_{MACS}$, такий же, як підроблений $PQPM_{MACS}$, або (ii) відповідь на зонд від звичайного хоста приходить (τ1), що має PRSPMACS не те саме, що підроблений RQPMACS або (iii) відповідь не надходить (і система повертається до стану x_1^1 на T_{10}^1), оскільки може статися так, що IP-адреса джерела запиту про пакет запиту RQPIPS не існує, і зловмисник не надсилає жодної відповіді на зонд. Як і в звичайному випадку, T_3^1 , T_5^1 іT_{10}^1 є некерованими переходами.

Нехай загориться перехід T_5^1 і приведе систему до x_6^1 . У x_6^1 є два варіанти – (i) відповідь на зонд від зловмисника приходить (T_6^1) з PRSPMACS так само, як і підроблений RQPMACS, або (ii) ніякої іншої відповіді на зонд не надходить (і система повертається до стану на T_{11}^1), оскільки може статися так, що зловмисник не надіслав жодної відповіді на зонд. Переходи T_6^1 і T_{11}^1 некеровані.

Якщо T_6^1 відбувається, система рухається до x_7^1, після чого T_8^1 відбувається миттєво. У ситуації атаки на запит на перевірку надійде більше одного запиту. Отже, після отримання більш ніж однієї відповіді зонда модель

повертається до початкового стану x_1^1 шляхом переходуT_8^1; T_8^1 неконтрольований.

Подібним чином можна пояснити іншу послідовність станів із x_3^1 (x_4^1 ,x_5^1 , x_1^1).

Можна відзначити, що модель не фіксує, яка реакція зонда (PRSP) відповідає нормі, а яка – зловмиснику. Модель лише позначає той факт, що є два відповіді з різними MAC-адресами; T_3^1, T_4^1 і T_5^1 , T_6^1 – це дві комбінації надходження відповідей з різними MAC-адресами. Далі, ці PRSP мають надійти протягом часу Treq після надсилання PRQP.

2.5 Діагностика для моделі DES підробки запитів ARP

У діагностику для моделі DES підміни запитів ARP не створюються переходи та стани з певних вузлів Fi. Діагностика оголошує атаку, коли досягається певний вузол Fi, оскільки оцінка містить стани лише з моделі атаки; подальша оцінка не потрібна, і можна зупинитися на певних вузлах Fi. Рисунок 2.7 ілюструє діагностик для моделі DES на рисунку 2.6.

У діагностиці вузли z1,z2,z3,z4(z5,z6,z7)є Fi-невизначеними (Fi-певними), і атака там не може (можна) бути виявлена. Можна відзначити, що існує два Fi-невизначених циклу. Наслідки таких невизначених циклів на діагностику атаки та усунення слідів для розриву таких циклів є наступним:

1. (z1,z2): Цей невизначений цикл виникає, якщо перехід T_2^1 від x_2^1 (T_2від x_2) не виконується, тобто PRQP не надсилається контролером. Іншими словами, без надсилання пакету проби немає різниці в послідовності подій ARP під атакою спуфінгу та нормальному стані, що робить атаку (збій Fi) недіагностованою. Цей невизначений цикл можна розірвати шляхом усунення сліду ((T_1' ,T_7')*) (оскільки T_7' і T_2' є керованими), що може бути досягнуто шляхом запуску події PRQP у стані x_2^1 (або x_2).

2. (z1,z2,z3,z4): Цей невизначений цикл виникає, якщо перехід T_4^1 від x_4^1 не виконується, тобто PRSP від справжнього хоста не надходить. Може

бути кілька причин, коли PRSP від справжнього хоста не надходить, а саме: (i) IP-адреса джерела в RSP (тобто PRQPIPD), що перевіряється, не існує.

Для цього PRQP (T_2^1) з неіснуючою IP-адресою призначення тільки зловмисник відповідає (T_3^1), оскільки тільки зловмисник знає про неіснуючу підроблену IP-адресу. (ii) IP-адреса джерела в RSP (тобто PRQPIPD), що перевіряється, є адресою самого зловмисника. (z1,z2,z3,z4) можна порушити, усунувши слід ($(T_1', T_2', T_3', T_{12}')^*$). Можна зазначити, що це неможливо, оскільки пер еходиT_4^1 і 12 є некерованими. Отже, якщо вихідна IP-адреса в IP-MAC RQP (T_1 або T_1^1) T_1 не існує або є адресою зловмисника, підробку не можна виявити.

Можна відзначити, що ці два випадки не призводять до складних ситуацій. Пов'язування IP-адреси неіснуючого хоста або IP-адреси самого зловмисника з іншим (помилковим) MAC-адресом призводить до переадресації трафіку, призначеного неіснуючому хосту або зловмиснику (на хост, який має пов'язаний помилковий MAC).

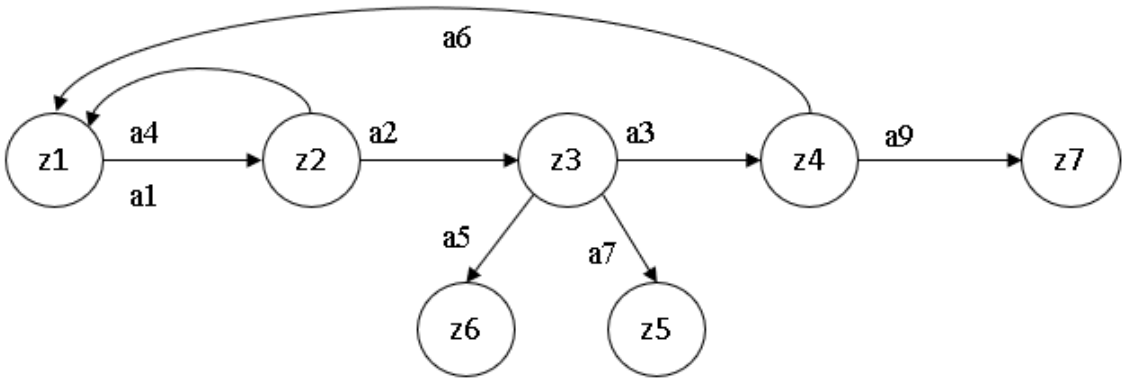


Рисунок 2.7 – Діагностик для моделі DES

Розглянемо випадок, коли виявлено атаку. Нехай діагностик досягає вузла z5 за послідовністю z1,z2,z3,z5 будучи Fi-певним O-вузлом, досягнутим при появі a7 (в силу T_5^1), оголошує атаку.

З Рисунку 2.7 можна помітити, що T_5^1 відповідає PRSP T^1 для PRQP протягом часу T_{req}, вихідний MAC якого не є таким, як вихідний MAC досліджуваного RQP.

2.6 Висновки

В розділі запропоновано моделювання процесу виявлення вторгнень в IT-інфраструктури. З цією метою було застосовано апарат дискретно-подійне моделювання (DES). Вказаний математичний апарат дозволив здійснити наступні дослідження, а саме розробити:

1. Дослідити механізм активного зондування.
2. Виконати моделювання обробки підроблених пакетів при атаці на IT-інфраструктуру.
3. Побудувати модель атак на IT-інфраструктуруз вимірюваністю.
4. Побудувати модель атак на IT-інфраструктуру з керованістю.
5. Побудувати моделювання відмов.
6. Здійснити перевірку (діагностику) а моделі.
7. ПобудуватиDES-модель атаки типу спуфінгу ARP.
8. Здійснити діагностику для моделі DES підробки запитів.

Побудовані моделі стануть основою для розроблення удосконаленого методу виявлення вторгнень в IT-інфраструктури.

3 УДОСКОНАЛЕНИЙ МЕТОД ВИЯВЛЕННЯ ВТОРГНЕНЬ В IT-ІНФРАСТРУКТУРИ

3.1Основи удосконаленого методу виявлення вторгнень в IT-інфраструктур

На основі запропонованих моделей, описаних в розділі 2, в роботі запропоновано удосконалений метод виявлення вторгнень в IT-інфраструктури. Метод ґрунтується на застосуванніDES-моделей, які характеризуються дискретним простором станів і певною динамікою, керованою подіями: в нормальних умовах, а також за кожного з умов атаки.

В основі методу лежить механізм оцінки стану (детектор), який спостерігає за послідовністю подій, створених системою, щоб вирішити, чи відповідають стани, через які система проходить, нормальній чи несправній моделі DES.

Важливим елементом методу є визначення стану атаки, яка розглядається як помилка, як було зазначено раніше.

Таким чином, для виявлення, наприклад, атаки NHU, спочатку система повинна бути змодельована як DES. Для цього представлена архітектура мережі, на якій схема має бути перевірена. Далі застосовується механізм активногозондування.

В загальному, метод містить наступні кроки:

1) Здійснення моніторингу мережі
2) Застосування механізму активного зондування
3) Перевірка правильності пакетів
4) Обробка підроблених пакетів
5) Моделювання DES в нормальних умовах та в умовах здійснення вторгнення
6) Застосування детектора

3.1.1 Архітектура мережі та її моніторинг

Для ілюстрації схеми використовується архітектура мережі, показана на рисунку 3.1. Хости A, B, C і D підключені до маршрутизатора R1. IDS відстежує мережевий трафік у своїй локальній мережі через свій дзеркальний порт. Дзеркальне відображення портів увімкнено на комутаторі, щоб IDS C мав копію всіх вихідних і вхідних пакетів з усіх портів. Маршрутизатори R2, R3 і R4 мають власні мережі з хостами X, Y і Z відповідно. Моніторинг NetFlow [47] включений у всіх інтерфейсах на маршрутизаторах першого стрибка, які безпосередньо підключені до IDS. Отже, IDS також відстежує ці пакети ICMP, отримані зовнішніми інтерфейсами R1 (тобто трафік ICMP від R2 і R3 до R1).

Усі системи, які контролюються, працюють; і працює, відповідають на запити IP у межах певного порогового інтервалу ticmp.

Дзеркальне відображення портів увімкнено на комутаторі в локальній підмережі IDS.

Системному адміністратору відомо про маршрутизатори (IP-адреси), безпосередньо підключені до мережі, і підтримує білий список таких IP-адрес, як IPGATEWAY.

SNMP з підтримкою аутентифікації [48] включений у всіх маршрутизаторах, які безпосередньо підключені до мережі. Отже, підробка повідомлень SNMP неможлива.

Припущення мережі зазначаються, оскільки підроблені пакети в атаках ICMP виглядають схожими на відповідні пакети, згенеровані за звичайних умов, використовується техніка активного зондування, щоб можна було розрізнити послідовність пакетів ICMP за нормальних умов та умов атаки. Всередині мережі IDS займається цим дослідженням.

Запропонована схема спирається на припущення, що зловмисник не може перешкодити справжньому хосту надіслати відповідь на запити, ініційовані IDS. IDS має два модулі, що працюють всередині.

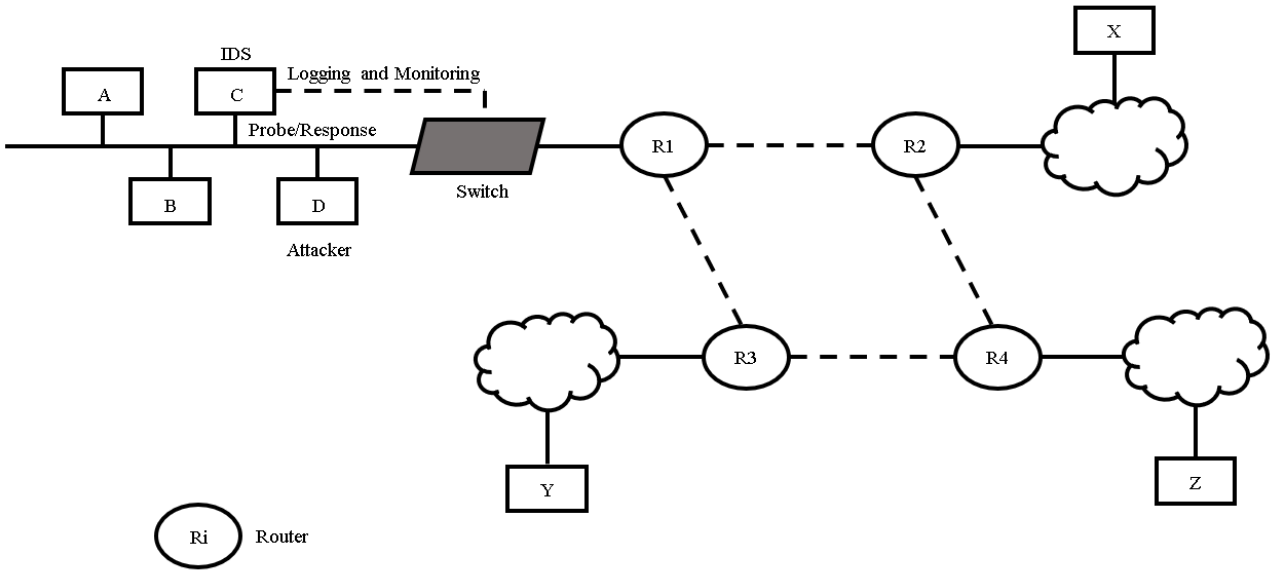


Рисунок 3.1 – Детектор DES для атаки NHU

Обробники: обробники надсилають пакети зондування, коли це потрібно, і повідомляють про події іншому модулю, тобто модулю детектора. Завдяки дзеркальному відображенню портів IDS отримує всі ICMP-пакети, згенеровані в мережі. Залежно від типу ICMP-повідомлення, що зустрічається, викликається один з обробників. Після виклику обробник обробляє пакети та генерує відповідні події.

Детектор: модуль детектора, який отримано в результаті процедури виявлення несправності, працює у фоновому режимі всередині IDS. Спостерігає за поданою йому послідовністю подій і вирішує, чи відповідає спостережувана послідовність умовам нападу чи нормальному стану.

3.1.2 Застосування механізму активного зондування

Техніка активного зондування пояснюється тут щодо IDS для атаки ICMP Net/HostUnreachable. Така ж методологія була застосована і для інших атак. Як уже обговорювалося раніше, під час атаки NetworkUnreachable зловмисник підробляє як маршрутизатор і надсилає помилковий пакет ICMP NetUnreachable, щоб перешкодити хосту-жертві підключитися до мережі призначення. Аналогічно під час атаки HostUnreachable зловмисник може

підробити пакет ICMP HostUnreachable і таким чином перешкодити хосту-жертві підключитися до хоста призначення. Після отримання пакетів ICMP Net/HostUnreachable, ICMP EchoProbes надсилається для виявлення атаки. Щоб допомогти в аналізі та розділенні справжніх повідомлень ICMP від підроблених, ведуться наступні таблиці.

Далі використовуються такі короткі позначення: IPS – IP-адреса джерела, IPD – IP-адреса призначення, NHU – мережа/хост недосяжний, NU – мережа недоступна.

1. SNMP_TABLE: SNMP-запит генерується для отримання різної інформації з маршрутизатора. Такі запити SNMP ініціюються IDS і називаються зондами SNMP. Надісланий зонд SNMP запитує значення ідентифікаторів об'єктів (theObjectIdentifiers– OID), зазначених у OID_LIST. OID_LIST– це кортеж, який містить один OID або набір OID залежно від вимог запиту SNMP. Значення, що відповідають кожному з OID у OID_LIST, повертаються в кортежі, позначеному OID_VALUE.

При отриманні SNMP-пакету деталі вносяться в SNMP TABLE. У цій таблиці зберігаються IP-адреса призначення, OID_LIST, OID_VALUEі мітка часу. Кожен запис у SNMP_TABLEпредставлений SNMPT. Ця таблиця містить чотири поля: IP-адреса призначення (SNMPTIPD), список ідентифікаторів об'єктів OID List (SNMPTOID_LIST), список значень OID Value (SNMPTOID_VALUE) і мітка часу (SNMPTT).

2. LOG_TABLE: Щоразу, коли виявляється атака, тип, код, IP-адреса джерела та IP-адреса призначення підробленого пакету зберігаються в цій таблиці разом із міткою часу, що стосується моменту виявлення спроби. Це історія спроб атаки і діє як файл журналу. Таблиця має п'ять полів і позначається LOGT. Його поля: тип (LOGTtype), код (LOGTcode), IP-адреса джерела (LOGTIPS), IP-адреса призначення (LOGTIPD) і мітка часу (LOGTT).

3. REDIRECT_table: поля: IPS (REDIRECTTIPS), Переважний шлюз (REDIRECTTprefgw), код (REDIRECTTcode), IPD (REDIRECTTIPD), REDIRECTTtimeofsending, REDIRECTTcount, REDIRECTTTTL,

$REDIRECTT_{count}$ і $REDIRECTT_{TTL}$ оновлюються пізніше, коли приходить зонд.

$(TableName)_{MAX}$ представляє максимальну кількість елементів у таблиці в певний момент часу.

3.1.3Перевірка правильності пакетів

IDS, як уже згадувалося в припущеннях, отримує копію всіх вхідних і вихідних пакетів з усіх портів, коли ввімкнено дзеркальне відображення портів. На IDS виконуються деякі початкові перевірки, щоб оптимізувати алгоритми обробки.

Під час ідентифікації пакетів ICMP виконується перевірка, щоб з'ясувати, чи є IDS неправильно сформованими чи ні. Якщо IDS неправильно сформовані, генерується сигнал тривоги, і відразу ж скидаються ще до виклику відповідних обробників.

Крім того, на цьому етапі можна перевірити відповідність пакетів ICMP деяким з уже існуючих сигнатур для атак ICMP, таких як *ping o ofdeath ma smur f*. У разі смерті ping сума зсуву IP і довжини даних буде більше 65535 байт. Якщо цей підпис збігається, системний адміністратор негайно повідомляє про атаку. Повідомлення ICMP Echo перевіряються на наявність підпису атаки smurf. Атака smurf спрямована на передачу запитів ICMP Echo з підробленим IP-адресом жертви як адресою джерела. Отже, якщо виявиться, що IP-адреса призначення є широкомовним доменом 255.255.255.255, можна виявити smurf. Аналогічно, інші сигнатури для помітних атак можна перевірити на цьому етапі, щоб уникнути тривалої обробки.

З метою оптимізації є дві глобальні змінні, які використовуються в усіх описаних алгоритмах – CONGESTIONSTATEтаICMPCOUNTER.

1. CONGESTION_STATEвикористовується для позначення стану модуля перевірки перевантаженості.

CONGESTION_STATEможе бути True, False або Unspecified.

CONGESTION_STATEвстановлюється як Unspecified, якщо модуль перевірки заторів не був викликаний протягом t_congestion. Перш ніж викликати модуль перевірки перевантаженості, спочатку перевіряється CONGESTION_STATE. Модуль перевірки перевантаження викликається лише якщо його стан Unspecified. СТАН ЗАКРІЧЕННЯ встановлюється як True або False залежно від результатів, отриманих модулем перевірки перевантажень. В іншому випадку безпосередньо використовує значення змінної CONGESTION_STATE. Це значно економить накладні витрати на обробку.

2. Крім того, щоб уникнути атаки затоплення, IDS підтримує ICMP_COUNTER, який може бути збільшений під час отримання повідомлень ICMP. Коли це значення лічильника досягає порогового значення NICMP протягом часу потоку t_{flood}, виявляється аномалія через раптове збільшення кількості пакетів і генерується сигнал IDS ALERT. Отримавши цей сигнал, IDS переходить у режим попередження та припиняє обробку пакетів, що переповнюються. Коли швидкість передачі пакетів повертається до нормального рівня, звичайна обробка всіх ICMP-пакетів IDS продовжується.

Розглянемо процес виявлення атаки ICMP NHU. Інші алгоритми, які використовуються для виявлення інших атак ICMP, будуть обговорені пізніше. Техніка зондування, що використовується для виявлення атаки NHU, використовує чотири основні обробники, а саме CONGCHECK(), NHUHAND(), ECHOREPLYHAND() і EXPHAND(). Ці обробки детально описані в Алгоритмі 3.1, Алгоритмі 3.2, Алгоритмі 3.3 і Алгоритмі 3.4 відповідно. Алгоритм 3.1 описує CONGCHECK(), тобто модуль перевірки перевантаженості, який використовується для виявлення заторів у шлюзі. CIP, CTYPE і CCODE використовуються для ідентифікації пакета ICMP, який зустрічається і для якого викликається CONGCHECK().Використовує запити SNMP для отримання інформації, пов’язаної з перевантаженням. Високе використання пропускної спроможності пояснює перевантаження. Отже, наявність перевантажень можна зробити висновок з оцінки використання пропускної здатності інтерфейсу. Використання пропускної спроможності базується на трьох параметрах – ifInOctets – кількість октетів, що вводяться в інтерфейс, ifOutOctets – кількість

октетів, що виходять з інтерфейсу, та ifSpeed – швидкість інтерфейсу в цей момент. Таким чином, SNMP-зонди надсилаються для кортежу OID LIST (SNMPOID_LIST) з трьома полями
<ifInOctets,ifOutOctets,ifSpeed>.

Три поля в кортежі OID_VALUE (SNMPOID _VALUE), тобто

<OID_ VALUE1,OID_ VALUE2,OID_ VALUE3>ініціалізуються на NULL. Значення, що відповідають трьом OID, повертаються в кортежі OID_VALUE, тобтоSNMPOID_VALUE.

Алгоритм 3.1 пояснює етапи механізму виявлення перевантажень. Алгоритм 3.1 приймає в якості параметра CIP (IP-адреса шлюзу, який перевіряється на перевантаження), CTYPE (тип повідомлення ICMP), CCODE (код повідомлення ICMP). Щоб виміряти використання пропускної спроможності, візьміть два набори значень ifInoctets і ifOutOctets з інтервалом часу tsnmp секунд. Ці значення вказуються як ifInOctetsold , ifInOctetsnew , ifOutOctetsold та ifOutOctetsnew. Потім ці значення підставляють у формулу для оцінки використання пропускної здатності і обчислюють результат BandwidthUtilization. Тепер, якщо BandwidthUtilizationбільше, ніж зазначений ThresholdBandwidth, STATE_CONGESTION повертається як True; в іншому випадку це False. Події CON і NOCON позначаються відповідним чином. Також підказуються CTYPE і CCODE. Ця перевірка перевантаження зазвичай виконується для інтерфейсу, безпосередньо підключеного до підмережі.

Алгоритм3.1 – CONGCHECK(CIP,CTYPE,CCODE)

Input:IP-IPaddressofhost,typeofICMPpacket,codeofICMPpacket,SNMP
TABLE,CONGESTIONSTATE
Output:IntimatecongestioninnetworkasCONandnocongestionasNOCON
SendSNMPQuerytoCIPforSNMPOIDLIST;
OnreceivingtheSNMPresponse,storethevaluesin<VAL1,VAL2,VAL3>correspond-
ingtoSNMPOIDVALUE;
AddCIP,SNMPOIDLIST,SNMPOIDVALUEandτtoSNMPTABLE;

$ifInOctets_{old}$=VAL1;

$ifOutOctets_{old}$=VAL2;Waitfort_{snmp}seconds;

SendSNMPQuerytoCIPfor$SNMP_{OIDLIST}$;

OnreceivingtheSNMPresponse,storethevaluesin<VAL1,VAL2,VAL3>correspond-

ingto$SNMP_{OIDVALUE}$;

AddCIP,$SNMP_{OIDLIST}$,$SNMP_{OIDVALUE}$andτtoSNMPTABLE;

$ifInOctets_{new}$=VAL1;

$ifOutOctets_{new}$=VAL2;ifSpeed=VAL3;

$\Delta ifInOctets = ifInOctets_{new} - ifInOctets_{old}$;

$\Delta ifOutOctets = ifOutOctets_{new} - ifOutOctets_{old}$;

$$BandwidthUtilization = \frac{\max(\Delta ifInOctets, \Delta ifOutOctets) * 8 * 100}{\Delta seconds * ifSpeed};$$

if (BandwidthUtilization ≥ ThresholdBandwidth) then

CONGESTION STATE = True;

Intimate CON,CIP,CTYPE,CCODE;

else

CONGESTION STATE = False; Intimate NOCON,CIP,CTYPE,CCODE;

endif

Exit;

Алгоритм 3.2 обробляє всі повідомлення мережі та повідомлення про недоступні хости в мережі. Передбачається, що повідомлення NetUnreachable надходять від шлюзу. Отже, для будь-якого повідомлення NetUnreachable NHU першим кроком є перевірка, чи збігається його вихідний IP NHUIPS з IPGATEWAY. Якщо є невідповідність, то виявляється підробка, і запис робиться в таблиці LOGT. В іншому випадку, якщо пакет NHU призначений для IDS, це означає, що це відповідь на зонд, надісланий IDS. У цьому випадку подія NHUPRSP повідомляється модулю детектора і викликається модуль CONGCHECK(). Для інших повідомлень Net/HostUnreachable IDS надсилає

ICMP EchoRequestProbe IPRQP до NHUIPS і викликає модуль EXPHAND() з NHUIPS, NHUtype, NHUcode, переданими як параметр. Блок-схема алгоритму 3.2 наведена на рисунку 3.2.

Алгоритм 3.2 – NHUHAND()

Input:NHU-ICMPNet/HostUnreachablepacket

Output:IntimatereceiptofNHU,NHUPRSPandsendingofICMPEchoprobereques
t

IPRQP

ifNHU$_{IPS}$notinIP$_{GATEWAY}$then

status←spoofed;

add{NHU$_{type}$,NHU$_{code}$,NHU$_{IPS}$,NHU$_{IPD}$,timeofreceipt}inLOGT;

elseif NHUIPD == IP(IDS) thenIntimatereceiptof NHUPRSP;
Waitforticmptime;

call CONGCHECK(IPGATEWAY, NHUtype, NHUcode);

else

Intimatereceiptof NHU;

Send ICMP EchoProbeRequest IPRQP to NHUIPS from IDS;
Intimatesendingofthe ICMP echoproberequest IPRQP; EXPHAND(NHUIPS,
NHUtype, NHUcode);

endif

Exit;

Алгоритм 3.3 описує ECHOREPLYHAND().Приймає як вхідні дані всі повідомлення ICMP Echo-відповіді мережі, але обробляє лише ті пакети, які призначені для IDS. Інші повідомлення з ехо-відповіддю відхиляються, оскільки не допомагають виявити будь-яку атаку ICMP. Блок-схема алгоритму 3.3 наведена на рисунку3.3. У разі виявлення атаки NHU, відповідного запису в REDIRECTT не буде, тому буде сповіщено IPRSP.

Алгоритм 3.4 описує алгоритм EXPHAND(). Блок-схема алгоритму 3.4 наведена на рисунку 3.4. Приймає в якості вхідних даних IP хоста (EIP), тип (ETYPE) і код (ECODE) пакету ICMP, для якого викликається. Після виклику обробника запускається годинник.

Алгоритм 3.3 – ECHOREPLYHAND()

Input:IPRSP-ICMPEchoReplypacket
Output:IntimatereceiptofIPRSP,IPRSPD,IPRSPP
ifIPRSP$_{IPD}$==IP(IDS)then
if IPRSP$_{IPS}$==REDIRECTT$_{IPS}$[i] AND
IPRSP$_{IPS}$==REDIRECTT$_{prefgw}$[i] AND
timeofreceiptofIPRSP- REDIRECTT$_{timeofsending}$[i]<ticmp,forsomei, 1 ≤ i ≤
REDIRECTT$_{MAX}$then
REDIRECTT$_{count}$[i]=REDIRECTT$_{count}$[i]+1;
ifREDIRECTT$_{count}$[i]>1then
status←spoofed;
add{5,REDIRECTT$_{code}$[i],IPRSP$_{IPS}$,REDIRECTT$_{IPD}$[i],timeofreceipt}inLOGT;
elseif REDIRECTTTTL[i] > IPRSPTTL then
Intimatereceiptof IPRSPP; Waitforticmptime;
IntimateTTLde f ;
else
Intimatereceiptof IPRSPP; Waitforticmptime;
IntimateTTLpre f ;
call CONGCHECK(IPGATEWAY, 5,REDIRECTTcode[i]);
endif
elseif IPRSPIPS == REDIRECTTIPS[i] ANDtimeofreceiptof IPRSP -
REDIRECTTtimeo f sending[i] <ticmp, forsome i, 1 ≤ i ≤ REDIRECTTMAX then
REDIRECTTcount[i] = REDIRECTTcount[i] + 1;
ifREDIRECTTcount[i] > 1 then
status ← spoo f ed;

add {5, REDIRECTTcode[i], IPRSPIPS, REDIRECTTIPD[i], timeo f receipt} in LOGT;
else
Intimatereceiptof IPRSPD; Waitforticmptime;
Send ICMP ProbeRequest IPRQPP toREDIRECTTpre f gw[i] from IDS;
add {REDIRECTTpre f gw[i], REDIRECTTcode[i], REDIRECTTpre f gw[i], REDIRECTTIPD[i], timeo f sending, 0, IPRSPTTL} in REDIRECTT;
Intimatesendingofthe ICMP proberequest IPRQPP;
Callexpiryhandler EXPHAND(REDIRECTTpre f gw[i], 5, REDIRECTTcode[i]);
endifelse
Intimatereceiptof IPRSP;
endifendifExit;

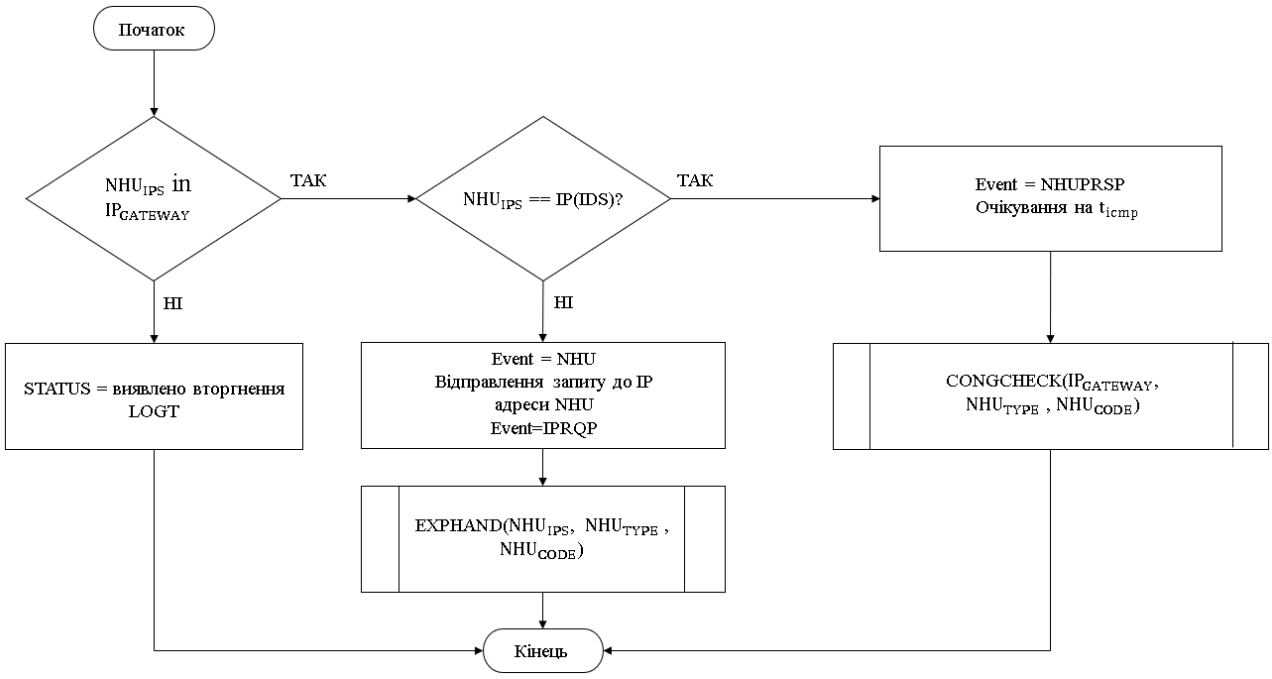


Рисунок 3.2 – Блок-схема для обробника недоступних мереж/хосту

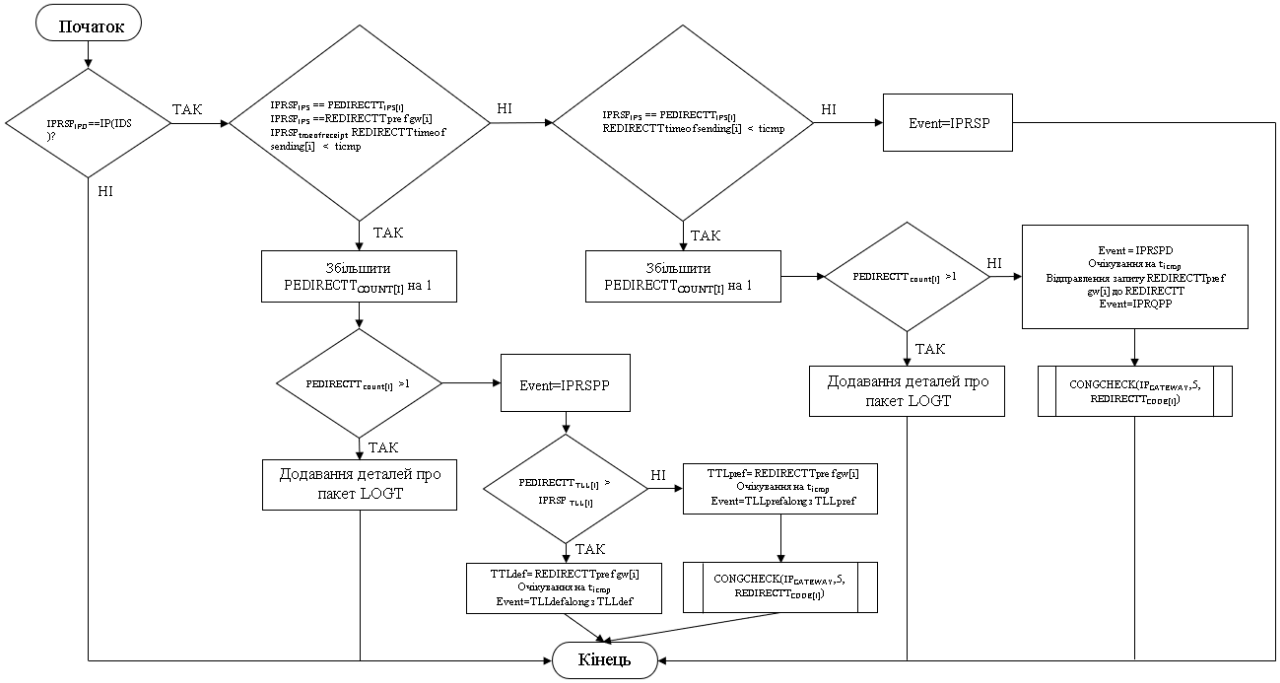


Рисунок 3.3 – Блок-схема для обробника exo-відповіді

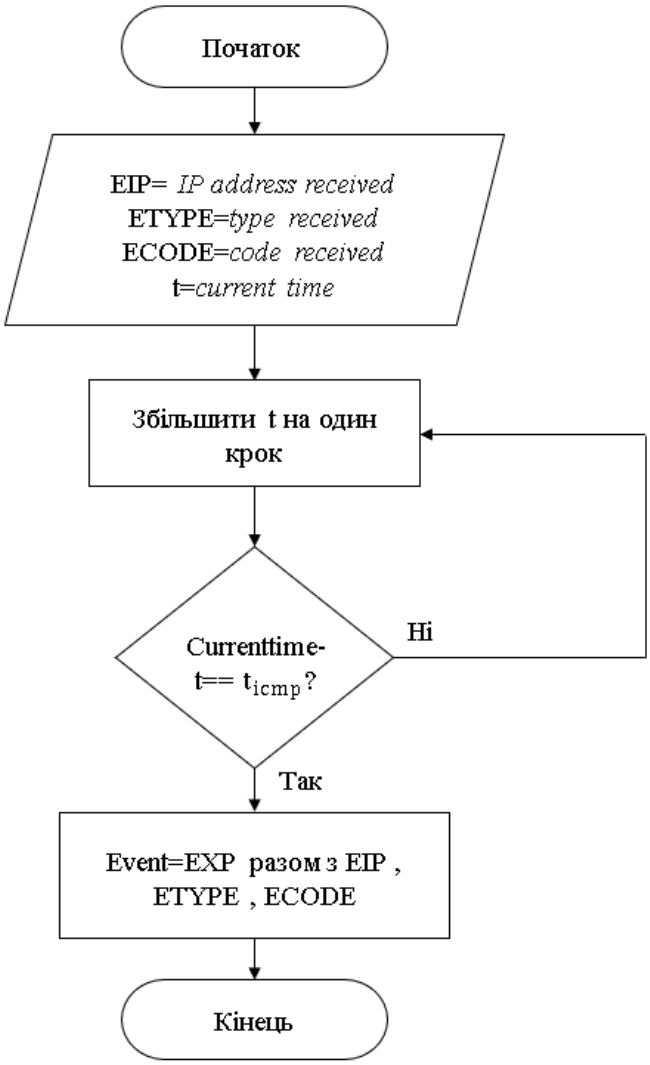


Рисунок 3.4 – Блок-схема для EXPHAND

Після того, закінчується часticmp, сповіщається подія EXP і повертається IP-адреса, тип і код, отримані як параметр.

Алгоритм3.4 – EXPHAND(EIP,ETYPE,ECODE)

Input : IP addressof a host, typeof ICMP packet, codeof ICMP packet.

Output: Intimate EXP andreturn EIP. t ← currenttime;

whiletruedo

clockticksupbyonestep;

ifcurrenttime − t == ticmp (valueofticmpispredefined) then

Intimate EXP, EIP, ETYPE and ECODE;

endif

endwhile

3.2Обробка підроблених пакетів

Розглянемо модель обробки підроблених пакетів на прикладі NHU.

Крім того, цей приклад також підкреслює різницю в послідовностях пакетів ICMP (після активного тестування) у випадку спуфінгу та звичайного сценарію.

На рисунку 3.5 показана послідовність пакетів (позначена порядковими номерами пакетів).

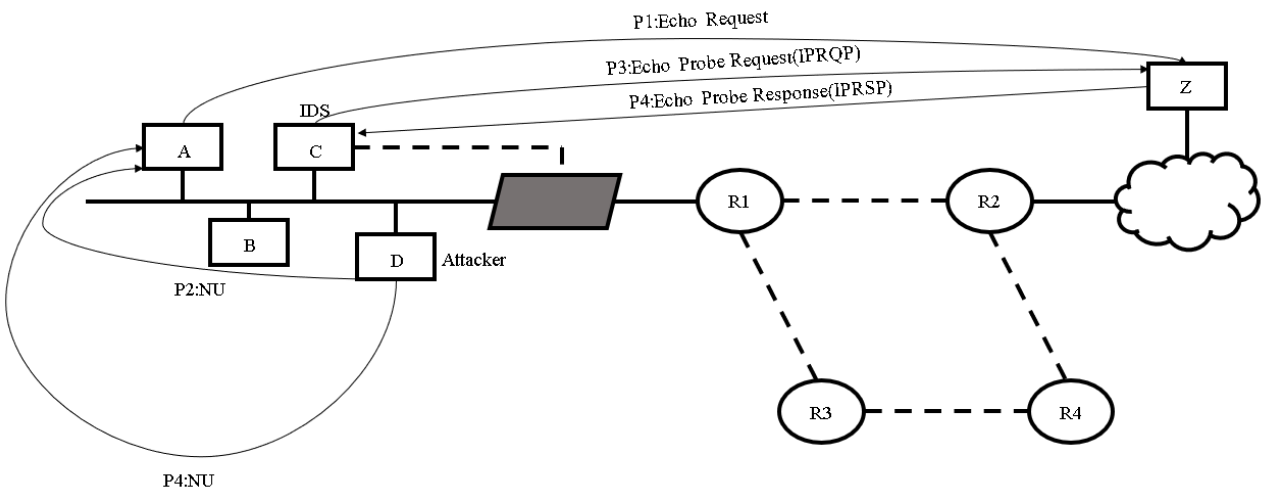


Рисунок 3.5 – Приклад для мережі/хост недоступний

Послідовність пакетів (P1): припустимо, що A хоче спілкуватися з хостом Z в іншій підмережі. Пінгуєхост Z і надсилає йому ICMP-ехо-запит.

Послідовність пакетів (P2): припустимо, що хост D є зловмисником, який хоче перешкодити A спілкуватися з Z. Таким чином, D відповідає на цей запит Echo, надсилаючи підроблений пакет мережі/хост недосяжний (NHU) хосту A.

Послідовність пакетів (P3): IDS отримує копію цього пакету NHU (P2) і в результаті викликається NHUHAND(). У справжньому випадку маршрутизатор R1 відправив би пакет NHU. Однак у випадку атаки D підробляє пакет так, ніби IDS надходить від R1. Таким чином, NHUIPS присутній в IPGATEWAY. Призначенням пакету є A, а не IDS. Тому перші дві умови обробника не збігаються. Таким чином, обробник надсилає запит ICMP EchoProbe до D і повідомляє про надсилання зонда IPRQP. Крім того, виконується виклик EXPHAND(), щоб гарантувати, що обробляються лише відповіді на зонд, які надходять протягом часу ticmp. Це дослідження створює різницю в послідовностях пакетів між нормальними умовами та умовами атаки.

Послідовність пакетів (P4): протягом часу ticmp Z відповість на запит ICMP Probe (P3) ICMP EchoReply. Ця подія допоможе детектору ідентифікувати атаку.Послідовність пакетів ($P4^1$): тепер існує ймовірність того, що відповідь не надійде в межах періодаticmp. Замість цього отримується ще одна недоступна мережа. Обробник NHU викликає CONGCHECK(), щоб отримати статус перевантаженості. Якщо перевантаження не існує, це означає, що мережа Z дійсно недоступна, інакше висновку не буде досягнуто. Модуль детектора приймає рішення та повідомляє системного адміністратора у разі атаки.

3.3 Моделювання DES в нормальних умовах та в умовах здійснення вторгнення

Модель DES, що використовується для представлення системи в звичайних сценаріях і сценаріях атаки, являє собою п'ять кортежів (X, X0, Σ, V, δ), де:

X – множина станів, X0 ⊆ X – множина початкових станів, Σ – множина подій, V – набір змінних моделі. Кожен елемент vi з V (V = {v1, v2, · · · , vn}) може приймати значення з області Di, f5 – множина переходів. Перехід τ ∈ δ – це п’ять кортежів (x, x^1, σ, check(V), assign(V)), де x – вихідний стан, x^1– стан призначення, σ – подія (на якій відбувається перехід). спрацьовує), check(V) – це булева кон’юнкція рівностей підмножини змінних у V, а assign(V) – це підмножина змінних моделі та присвоєнь зі значеннями з відповідних областей.

Представлення GN : (XN , X0N , ΣN , f5N , VN) використовується для моделі DES за звичайних умов, а GFi : (XFi, X0Fi, ΣFi, f5Fi, VFi) використовується для моделі DES під час відмови (атаки) Fi. Тут і використовується для позначення і-го типу відмови, оскільки може бути кілька збоїв і більше однієї моделі атаки DES, що відповідає їм. У цій главі для кожної моделі системи є один тип моделі відмови, позначений Gf1, але при описі запропонованих алгоритмів розглядаються численні відмови.

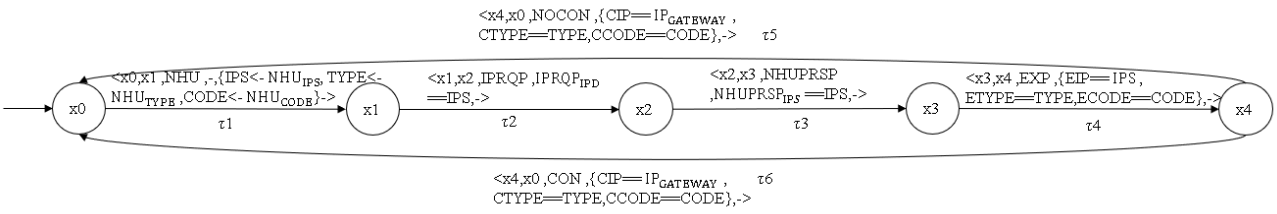


Рисунок 3.6 – Модель DES в нормальному стані

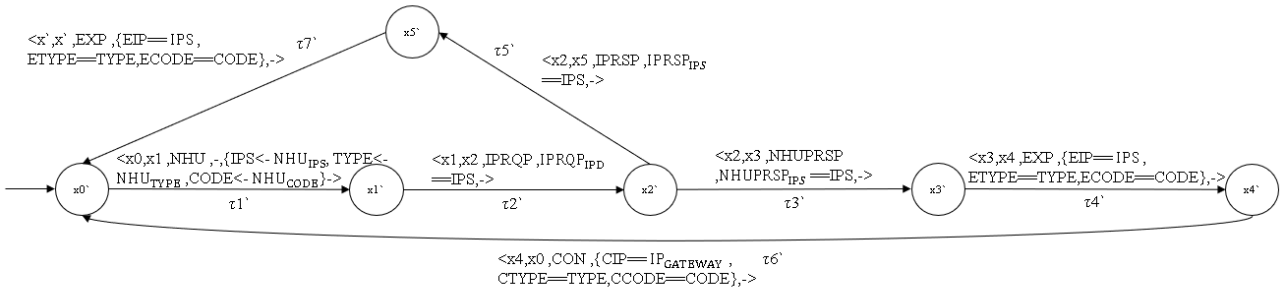


Рисунок 3.7 – Модель DES в умовах атаки

Тепер демонструється моделювання мережі DES під час атаки NHU. На рисунку 3.6 показано звичайну модель DES GN мережі. На рисунку 3.7 показана модель DES GF1 за сценарієм атаки NHU. Можна відзначити, що в моделі DES немає остаточного стану, оскільки трафік ICMP в мережі є безперервним (або оновлюваним) процесом (ніколи не зупиняється, поки мережа працює). Є деякі стани оновлення, з яких відбувається перехід до початкового стану(ів). Таким чином, система працює.

Терміни, необхідні для моделювання DES, кількісно визначаються таким чином:

Σ = {NHU, IPRQP, NHUPRSP, IPRSP, EXP, CON, NOCON}. Набір станів X показано на рисунку 3.6 та рисунку 3.7. Стани без простих чисел відповідають нормальній ситуації, а стани з одними простими числами позначають стан атаки відповідно. Початкові стани (x0,x1) також показані на рисунках 3.6 і 3.7. Набір змінних моделі V = {IPS, TYPE, CODE}; IPS має домен як D1 = {y.y.y.y|1 ≤ y ≤ 255}; TYPE має область визначення як D2 = {y|0 ≤ y ≤ 7}; CODE має область визначення D3 = {y|8 ≤ y ≤ 15}. Переходи також показані на рисунках 3.6 та 3.7, переходи без простих чисел призначені для нормальної моделі, а окремі прості числа – для атаки. Можна відзначити, що для деяких полів у кортежі, що представляють переходи, є «−11. Якщо « − » призначено для check(V), то представляє умову TRUE, тоді як якщо « − » для assign(V), то представляє NOдію (тобто присвоєння) не потрібно. Огляд моделі DES для сценарію атаки NHU (Рисунок 3.7) пояснюється наступним чином.

Модель переходить у стан x_1^1 при спостереженні події NHU шляхом переходу T_1^1. Увімкнення T_1^1 залежить лише від NHU, а не від будь-яких змінних моделі; це зображено символом « − » на переході. Змінним моделі IPS, TYPE і CODE призначаються IP-адреси джерела, тип і код пакету ICMP NHU відповідно. Після цього в стані x_2^1 надсилається ICMP-зонд. Перехід T_2^1 увімкнено на IPRQP (відправляється на вихідний IP NHU, що розглядається); Відповідність IPRQP NHU визначається шляхом перевірки, чи відповідає змінна моделі IPS з IPRQPIPD. Після переходу T_2^1 є два варіанти; (i) або відповідь на зонд від справжнього хоста надходить (T_5^1) у вигляді exo-відповіді, або (ii) відповідь зонда (T_3^1) надходить у вигляді іншого пакету NHU. Тепер, у першому випадку, стає очевидним, що початковий пакет NHU був підроблений. T_7^1запускається подією EXP, яка вказує, що більше не потрібно чекати відповіді, і система оновлюється. У другому випадку, тобто коли NHUPRSP отримано як відповідь на зонд, початковий пакет NHU здебільшого є справжнім. Але існує ймовірність того, що в мережі є перевантаження, через яке справжній хост не може відповісти, а тим часом зловмисник підробив відповідь NHUPRSP (Net/Hostunreachablemessage з підробленим IP). Щоб вирішити цю проблему, використовується CONGCHECK(). Якщо немає перевантажень NOCON, робиться висновок, що повідомлення, безсумнівно, справжнє. Однак у разі події CON рішення стає неоднозначним. Таким чином, перехід T_8^1 додається до моделі атаки. T_8^1 переводить управління в початковий стан, подібно до T_7^1. Це відновлювальні переходи.

3.4 Застосування детектора

Після розробки моделей нормальної та атакуючої DES розроблено детектор, який може визначити, чи відповідає дана послідовність подій нормальному сценарію чи сценарію атаки. Детектор, по суті, є різновидом оцінювача стану, який оцінює, який стан перейти, коли йому передається подія. Нарешті, детектор оголошує атаку, коли досягається стан (детектора), оцінка

якого включає стани лише з моделі атаки, після завершення відстеження подій. Перед формальним обговоренням детектора вводяться такі визначення:

Приймемо два переходи T1 = (x_1, x_1^1 , σ1, check1(V), assign1(V)) і T2 = (x_2, x_2^1 , σ2, check2(V), assign2(V)) є еквівалентними, якщо σ1 = σ1 (та сама подія), check1(V) ≡ check2(V) (ті самі рівність для однієї і тієї ж підмножини змінних у V) і assign1(V) ≡ assign2(V) (та сама підмножина змінних моделі з однаковим призначенням).

Якщо T1 ≡ T2, то вихідні стани переходів еквівалентні, а також стани призначення, тобто $x_1 \equiv x_2$ і $x_1^1 \equiv x_2^1$.

Детектор представляється у вигляді орієнтованого графіка O =< Q, A > де, Q – набір станів детектора, які називаються O-станами, A – набір детекторних переходів, які називаються O-переходами.

Відтепер такі терміни, як переходи, стани і т. д. DES, називаються переходами моделі, станами моделі тощо, щоб відрізнити їх від детектора. Кожен O-стан q ∈ Q містить підмножину еквівалентних модельних станів, що представляють невизначеність щодо фактичного стану, а кожен O-перехід a ∈ A являє собою набір еквівалентних модельних переходів, що представляють невизначеність щодо фактичного переходу, що відбувається. Початковий O-стан включає всі початкові стани моделі. Після цього, для будь-якого O-стану q, O-переходи, що виходять з q, отримують наступним чином. Нехай f5q позначає множину модельних переходів із станів моделі x ∈ q. Нехай Aq– множина всіх класів еквівалентності f5q. Для кожного a ∈Aq створюється O-перехід, що містить усі модельні переходи в a. Тоді наступний O-стан для a будується як q^+= {x|x– стан моделі призначення τ ∈ a}. Наступник для O-станів, що визначає атаку, або нормальні сценарії не створюються. Цей процес повторюється до тих пір, поки не вдасться створити новий *O*-перехід.

Приймемо O-стан, який містить лише модельні стани, що відповідають нормальній ситуації, називається нормальним певним O-станом.

Приймемо O-стан, який містить лише модельні стани, що відповідають атакам, називається Attacksome O-state.

Приймемо O-стан, який містить модельні стани, що відповідають як

звичайній ситуації, так і ситуації атаки, називається невизначеним станом.

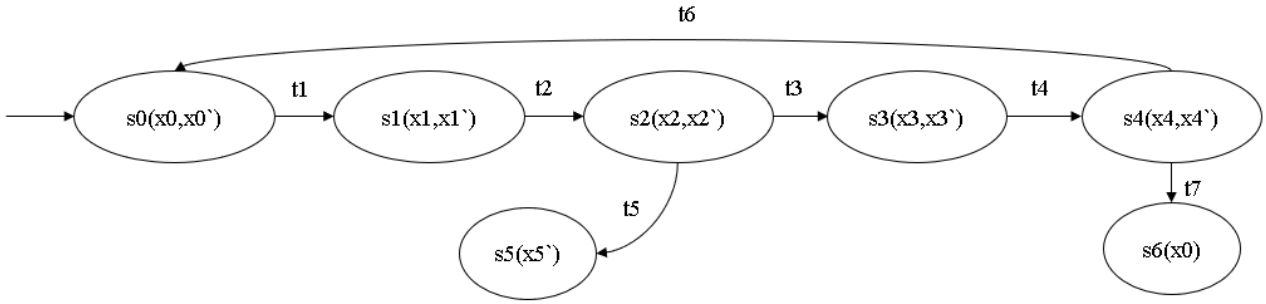


Рисунок 3.9 – I-детектор для моделі DES на рисунку 3.6 і рисунку 3.7

3.5 Застосування I-детектора

Попереднє визначення діагностичності вимагає відсутності невизначеного циклу атаки у всіх слідах, створених детектором *O*.

Послаблене визначення діагностичності, яке називається I-діагностичною, вимагає, щоб умова діагностики була дійсна лише в тих слідах, де атаки невизначений *O*-стан супроводжується індикаторним переходом, не в кожній трасі, що містить збій.

Таким чином, збій пов'язаний з одним або кількома переходами індикаторів.

Події, що ініціюють переходи індикаторів, є індикаторними подіями.

Якщо несправність сталася до індикаторної події, то цю несправність необхідно виявити протягом кінцевої затримки після її виникнення.

Таким чином, система I-діагностується, якщо після індикаторної події в жодному сліді немає циклу невизначених атаки *O*-станів.

Це розслаблене визначення допомагає виконати часткову діагностику системи на деяких шляхах, де визначені події індикаторів, навіть якщо система в цілому не піддається діагностиці.

I-детектор будується шляхом додавання функції маркування до орієнтованого графіка детектора O. Таким чином, I =< S, T, L > де, S – набір станів I-детектора, які називаються I-станами.

Цей набір такий самий, як і для станів детектора Q, T – набір I-детекторних переходів, які називаються I-переходами. Цей набір також такий же, як і для детекторних переходів A, L – функція маркування, яка використовується для позначення I-станів. L(s) = f − i, якщо s є невизначеним станом атаки, а траса від початкового стану, що веде до s, містить відповідну індикаторну подію. L(*s) = нічого*, інакше.

I-стан *s* називається станом *f-i*, якщо міткою стану s є *L(s) = f-i.*

Невизначений цикл *f-i* в I - це цикл, що складається лише з *f-i* станів. Це означає, що існує цикл як у звичайній моделі, так і в моделі атаки, які створюють однакові записи переходів, а цикл у моделі атаки має індикаторну

подію.

Система I-діагностується, якщо в I немає *f-i* невизначеного циклу. Наявність такого циклу означає, що не можна буде виявити виникнення несправності навіть у тих слідах, які містять індикатор, як система може продовжувати зациклюватися в цих станах вічно.

Тепер у разі виявлення детектора атаки NHU цю часткову діагностику можна зробити. Індикаторна подія вибирається як NOCON, оскільки цілком закономірний факт, що діагностика несправності неможлива в перевантаженій мережі. Подія NOCON означає, що CONCHECK() підтвердив, що в мережі немає перевантажень. Використовуючи цю індикаторну подію, будується I-детектор, як показано на рисунку 3.9. Процедура побудови подібна до детектора, за винятком належного маркування, як було визначено раніше. Щоб уникнути повторення, детальний опис цього опущено.

Можна помітити, що в I-детекторі на малюнку 3.9 немає f *-i* невизначеного циклу, хоча є атака невизначеного циклу $s0,s1,s2,s3,s4,s0$. У цьому невизначеному циклі атаки не відбувається переходу, ініційованого індикаторною подією *NOCON*. Тому це не порушує умови I-діагностичності. Несправність певної траси, $s0,s1,s2,s5$, як правило, діагностується. Слід, на якому відбувається індикаторний перехід $\tau 5$, є нормальною певною трасою $s0,s1,s2,s3,s4,s6$. Процес діагностики може продовжуватися за цими двома слідами згідно з визначенням, виключаючи невизначений цикл. Тепер обговорюється виявлення сценарію атаки та нормального сценарію для пакету NHU.

Сценарій атаки. Нехай детектор досягне стану $s5$за послідовністю $s0,s1,s2$. У станах $s1,s2,s3$атака або нормальний стан неможливо визначити, оскільки оцінка стану містить один нормальний стан моделі та один стан моделі атаки, наприклад, $s1$ має x_1і x_1^1. Оскільки $s5$ - це певний I-стан атаки, досягнутий при настанні $t5$ (за рахунок T_5^1), I-детектор оголошує атаку. З рисунка 3.7 можна помітити, що T_5^1 відповідає *IPRSP* для *IPRQP* (T_2^1) протягом часу ticmp. $t1$ - це I-перехід, що виходить від $s0$ і знаходиться на шляху, що веде до $s5$. Подія, що відповідає $t1$ (те саме, що і $T1/T_1^1$) є *NHU*, що вказує на

отримання підробленого пакету *NHU*. Отже, в $s5$ деталі цього пакету атаки реєструються в таблиці *LOGT*.

Нормальний сценарій. Нехай детектор досягає стану $s6$ за послідовністю $s0,s1,s2,s3,s4$(без нормальних/атаки певних станів). $s6$ є нормальним певним *I*-станом, досягнутим при появі *t7* (за рахунок T5), таким чином, *I*-детектор оголошує пакет, що перевіряється, як нормальний. З рисунка 3.6 можна помітити, що τ5 відповідає тому факту, що *NHUPRSP* прибув для *IPRQP* протягом часу *ticmp* і після цього не було виявлено перевантажень в мережі.

Сценарій циклічного циклу невизначених станів. *I*-детектор може зациклюватися в послідовності $s0,s1,s2,s3,s4,s0$(не включаючи нормальних/атакуючих певні *I*-стани). У цьому випадку атака не може бути виявлена протягом кінцевої затримки, навіть якщо вона вже відбулася. Але в цьому циклі не відбувається перехід індикатора (*NOCON*). Таким чином, умова діагностики може бути послаблена для цього сліду, не порушуючи загальне уявлення про I-діагностику.

3.6Здійснення часткової діагностики за допомогою зменшеного I-детектора

З часткової діагностики, зробленої в попередньому розділі, видно, що *I*-детектор, створений для атаки NHU, містить непотрібний шлях (тобто $s0,s1,s2,s3,s4,s0$), де діагностика неможлива. У великій складній системі може бути багато таких слідів, які не можна діагностувати, але *I*-детектор їх містить. Це забезпечує мотивацію для створення зменшеного *I*-детектора (званого *RI*-детектором), який містив би лише ті шляхи, де можна виконати діагностику несправностей. У цьому розділі запропоновано алгоритм побудови цього зменшеного детектора. Доведено, що час роботи цього детектора менше, ніж у *I*-детектора, і показано роботу *RI*-детектора для атак ICMP.

3.7 Застосування RI-детекторf

У детекторі на рисунку 3.9, $s0,s1,s2,s5$є слідом, де атака може бути підтверджена. Після переходуa_5 детектор вперше переходить з невизначеного стану до стану, що визначатиме атаку. Такий перехід називається AD-переходом (AttackDetectingTransition). Перехідa_5відповідає T_5^1 з моделі атаки, але AD-перехід не має відповідного переходу від нормальної моделі. Таким чином, за допомогою цих AD-переходів помилка завжди діагностується. Машина може бути розроблена для виявлення виникнення таких переходів AD. Ця машина називається *RI-детектор*.

У типовій складній системі кількість станів і переходів може бути дуже великою. Варто розробити RI-детектор, який може виявляти наявність атаки, відстежуючи лише AD-переходи замість усього набору переходів. Виявлення AD-переходу є двоетапною процедурою. На першому кроці *RI*-детектор перевіряє, чи переходить отриманий перехід систему в стан, з якого може виходити AD-перехід. Якщо це зробить, на наступному кроці RI-детектор перевіряє, чи є наступний перехід еквівалентним AD-переходу, що виходить з цього стану. Якщо це сталося, детектор робить висновок, що сталася несправність. Зауважимо, що набір AD-переходів синтезується заздалегідь. Також обидва кроки можна виконати, лише подивившись на систему та набір AD-переходів.

3.8Структура RI-детектора

Перед описом процедури побудови *RI*-детектора формально охарактеризовано AD-переходи та *RI*-детектор.

Перехід *a* детектора є *AD*-переходом, якщо вихідний стан *a* є невизначеним *O*-станом, а стан призначення a є певним O-станом атаки.

RI-детектор формально представляється як орієнтований граф

$$RI = < M, B, Ou>, \qquad (3.1)$$

Де*M*– множина станів, які називаються *RI*-станом.

B– набір переходів, які називаються *RI*-переходами.

Ou– вихідна функція, визначена як*Ou(b)* ⟶ *{Y, N},* де *b*– *RI*-перехід.

Кожен *RI*-перехід *b* є трьома кортежом (*s, t,* o_{val}), де *s* – *O*-стан, *t*– *O*-перехід, а o_{val}– вихідне значення, створене *Ou*; вихід *Y* відповідає несправності, тоді як *N* означає, що несправність не може бути підтверджена.

Взагалі в *RI*-детекторі є три типи станів - початковий стан $m0$, проміжний стан для кожного AD-переходу, $m1,m2,...,ml$(припускаючи f5*AD* = { $a1,a2,...,al$ } – набір AD-переходів) і єдиний кінцевий стан *mf* . Нехай T_i^1– відповідний перехід у моделі атаки для AD-переходу *ai*∈ f5*AD*. Всякий раз, коли модель атаки переходить у стан x^1, такий, що x^1 є вихідним станом переходу T_i^1, *RI*-детектор переходить у проміжний стан mi. Таким чином, відбувається перехід від m0 до mi, заданого (*стан джерела ofai, −, N*). «-»
позначає не турбується, що означає, що не потрібна перевірка переходу, здійсненого системою. Для інших переходів зациклюється на початковому стані з овалом = N . Тепер на m_i, якщо модель атаки переходить у стан x^1, такий, що x^1 є станом призначення переходу T_i^1, RI-детектор переходить до кінцевого стану *mf* . Перехід від m_iдоmfзадається як (*стан призначення ofai, ti, Y*). Якщо система виконує будь-який інший перехід, *RI*-детектор повертається до початкового стану *m0* з o_{val}=*N*. Як тільки детектор досягає стану *mf*, залишається там назавжди, що представляє собою постійний збій.

Як було зазначено раніше, для побудови *RI*-детектора спочатку необхідний набір AD-переходів. Цей набір AD-переходів можна отримати з детектора, але побудова всього детектора не потрібна. Більш уважний огляд показує, що потрібні лише невизначені стани та певні стани, які не входять до норми. Нормальна і атака моделі є вхідними. Набір AD-переходів отримано з них за допомогою Алгоритму 3.6.

У разі атаки NHU з рисунка 3.8 видно, що початковий стан $q0$ містить $\{x_1, x_0^1\}$ – невизначений стан. Він додається до набору станів *S*. З $q0$ після події

NHU переходить до іншого невизначеного стану $q1$, що містить $\{x_1, x_1^1\}$. Аналогічно переходить у стан $q2$ після цього. Усі ці невизначені стани додаються в *S*. Від $q2$ після переходу $a5$ виходить шлях, який веде до атаки на певний стан $q5$, що містить x_5^1. Отже, перехід $a5$ додається до множини AD-переходів τ_{AD}. Не існує іншого такого переходу з невизначеним станом джерела та визначальним станом призначення. Таким чином, коли алгоритм завершується, τ_{AD} містить один єдиний перехід $a5$, що має вихідний стан $q2$ і стан призначення $q5$. Після отримання цього набору τ_{AD} можна побудувати *RI*-детектор за допомогою Алгоритму 3.7.

Алгоритм 3.6 Findingthesetof AD transitions
Input: DES models GN forNormalandGFiwhere 1 ≤ i ≤ f forAttackconditions
Output: Thesetof AD transitions f5AD
Begin
s0 ← X0N ; /* initialstatesofnormalmodel */
FOR ALL i, 1 ≤ i ≤ f /* numberoffailuretypes */
s0 ← s0 ∪ X0Fi ; /* initialstatesofeachfailuremodels */
S ← s0; T ← φ;
FOR ALL s ∈ S whichareuncertainDo {
f5s ← {τ|τ∈ {f5N ∪ f5Fi∧sourcestateof τ ∈ s}; FindthesetofallequivalentclassesTs, of f5s; FOR ALL t ∈Ts {
s+ = {x|xisdestinationstateof τ ∈ t};
IF s+ isanuncertainstate THEN
S = S ∪ {s+};
IF s+ isattackcertain THEN
τAD ← τAD∪ t; T = T ∪ {t};
} /* Forall t ∈Ts */
} /* for s ∈ S whicharenotcertain */
End

Стани $m0$ і mf завжди будуть міститися *RI*-детектором. Отже, ці два стани додаються першими. Потім для кожного AD-переходу, наявного у f5AD, спочатку створюється стан m_i. Потім відповідні переходи додаються до і від $m0$ і mf *до* стану m_i, як пояснювалося раніше. Це робиться для кожного AD-переходу. Нарешті, до $m0$ і mf додаються переходи з власними циклами з відповідними виходами. На цьому побудова *RI*-детектора закінчена.

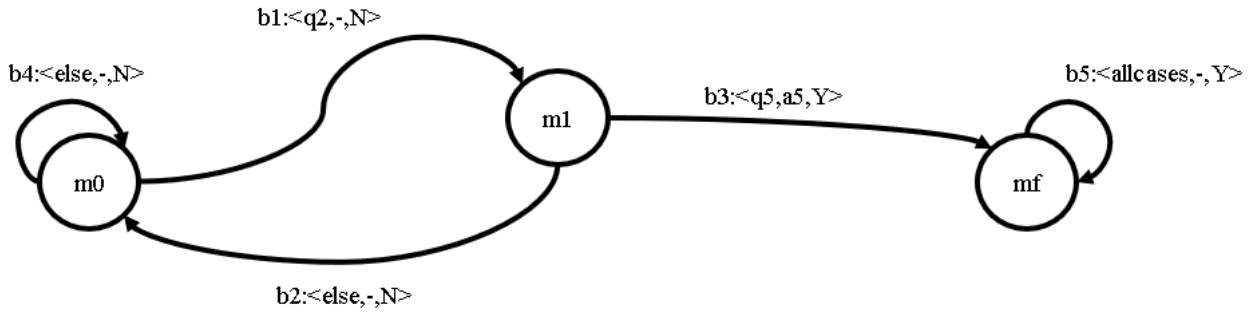


Рисунок 3.10 – RI-детектор для атаки NHU

Сценарій виявлення атак: RI-детектор для атаки NHU показаний на рисунку 3.10. Цей *RI*-детектор призначений для виявлення виникнення AD-переходу $a5$. Перехід починається зі стану $m0$, досягає стану $m1$ шляхом переходу $b1$ щоразу, коли система переходить у стан x_2^1, тобто вихідний стан T_5^1, що міститься в $a5$. Значення не турбується в перевірці переходу означає, що цей перехід приймається як і коли наступний стан оцінюється як x_2^1, без огляду на вхідний перехід.

3.9 Висновок

В даному розділі було запропоновано удосконалений метод виявлення вторгнень в IT-інфраструктури, що ґрунтується на застосуванні DES-моделей, які характеризуються дискретним простором станів і певною динамікою, керованою подіями в нормальних умовах, а також за кожного з умов атаки. В основі методу лежить механізм оцінки стану (детектор), який спостерігає за

послідовністю подій, створених системою, щоб вирішити, чи відповідають стани, через які система проходить, нормальній чи несправній моделі DES, і далі застосовується механізм активного зондування.

Також в розділі було представлено, наприклад, виявлення вторгнення атакою NHU.

Метод містить наступні кроки: моніторинг мережі, застосування механізму активного зондування, перевірка правильності пакетів, обробка підроблених пакетів, моделювання DES в нормальних умовах та в умовах здійснення вторгнення, застосування детектора.

Застосування методу дозволить підвищити достовірність та ефективність виявлення вторгнень в IT-інфраструктури

4 ПРОГРАМНО-ТЕХНІЧНИЙ ЗАСІБ ВИЯВЛЕННЯ ВТОРГНЕНЬ В ІТ-ІНФРАСТРУКТУРИ

4.1 Експериментальні дослідження програмно-технічного засобу виявлення вторгнень в ІТ-інфраструктури

В результаті реалізації удосконаленого методу було реалізовано програмно-технічний засіб виявлення вторгнень в ІТ-інфраструктури.

Стандартними контрольними показниками для програмно-технічного засобу виявлення вторгнень в ІТ-інфраструктури є точність, швидкість виявлення, споживання ресурсів тощо. Оскільки деякі додаткові повідомлення ARP генеруються для перевірки, розглядається додаткова статистика трафіку також як еталон разом із точністю, швидкістю виявлення та споживанням ресурсів. Далі наведено результати в різних ситуаціях мережевого трафіку.

Під час атаки підробки запитів ARP зловмисник надсилає (шляхом широкомовної передачі) пакет запиту ARP з фальсифікованою парою IP-MAC (у полях адреси апаратної адреси-відправника протоколу відправника). Хост з IP-адресою, вказаною в полі «адреса цільового протоколу» пакета запиту, оновлює свій кеш із неправильним з’єднанням IP-MAC.

Наприклад, нехай є три хости A,B і D; A і B є справжніми, а D є нападником. Під час підробки запиту (націлювання на A) зловмисник D надсилає підроблений пакет запиту ARP з адресою цільового протоколу як IP(A) і апаратною адресою відправника-адресою протоколу відправника як IP(B)-MAC(D); A оновить свій кеш MAC-адресою D, що відповідає IP-адресі B. Це призведе до того, що пакети, призначені для відправки B (A), будуть надіслані D.

З іншого боку, якщо D надсилає підроблений пакет запиту ARP з IP(B)-MAC(X) (де MAC(X) - це MAC-адреса неіснуючого хоста), тоді всі пакети, призначені для відправлення до B (A), будуть втрачені. Спуфінг відповіді ARP схожий на підробку запитів ARP.

Однак у випадку підробки відповіді використовується пакет відповіді ARP. Для стислості проілюструємо запропоновану схему лише для атаки підробки запитів; те ж саме стосується атак підробки відповіді.

4.2 Архітектурапрограмно-технічний засіб виявлення вторгнень в IT-інфраструктури тестового стенду

Оскільки ARP не має стану, послідовність пакетів ARP у звичайних і підроблених умовах однакова. Таким чином, для виявлення спуфінгу потрібен механізм, який може створювати різницю в послідовності пакетів за нормальних умов і умов атаки; це поняття використовується в запропонованій IDS.

Основна архітектура мережі, що контролюється за допомогою IDS, показана на рисунку 4.1. Як показано на рисунку, IDS містить діагностик DES і контролер контролю.

програмно-технічний засіб виявлення вторгнень в IT-інфраструктури (фактично реалізована IDS) отримує весь трафік завдяки дзеркальному відображенню портів і надсилає запити на перевірку справжності запитів і відповідей ARP.

Той факт, що ці пробні запити створюють необхідну різницю в послідовності пакетів, ілюструється на прикладі, наведеному нижче.

Діагностика пасивно відстежує всі події, а саме запити ARP, відповіді APR, відповіді на зонд тощо (неконтрольовані події) і визначає, чи відбулася атака (рисунок 4.1).

Приклад сценарію підробленої відповіді з зондуванням.

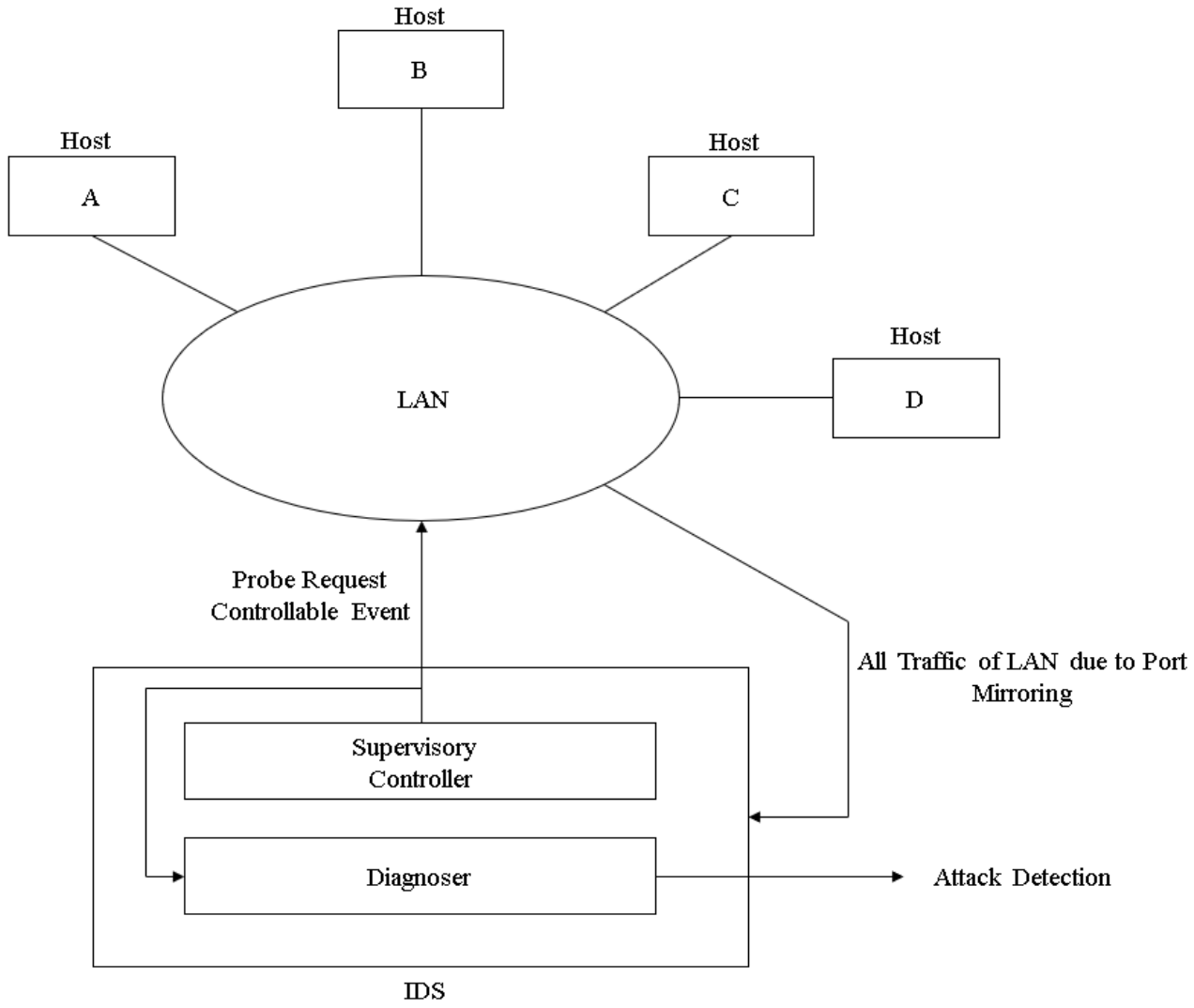


Рисунок 4.1 – Базова архітектура тестового стенду з IDS

Розглянемо мережу (рис. 4.2), що має два справжніх хоста A і B, D є зловмисником, а програмно-технічний засіб виявлення вторгнень в IT-інфраструктуримає IP=IP(E), MAC=MAC(E).

На рисунку 4.2показана послідовність пакетів (позначена порядковими номерами пакетів), що вводяться, коли зловмисник D надсилає підроблений запит як «IP(B)-MAC(D)» на широкомовну адресу, запитуючи MAC-адресу хоста A та її перевірку (за допомогою активного зонда).).

У послідовності пакетів 1 D надсилає запит ARP *RQP* на широкомовну адресу, запитуючи MAC-адресуA. Після отримання цього повідомлення запиту від D, A оновлює свій кеш парою IP-MAC як IP(B)-MAC(D), а потім відповідає D власною MAC-адресою. Можна зазначити, що за звичайних умов «запит B до

A з парою IP-MAC :IP(B)-MAC(B)» та умова атаки «запит D до A з парою IP-MAC :IP(B)-MAC(D)», послідовність ARP-пакетів не змінюється.

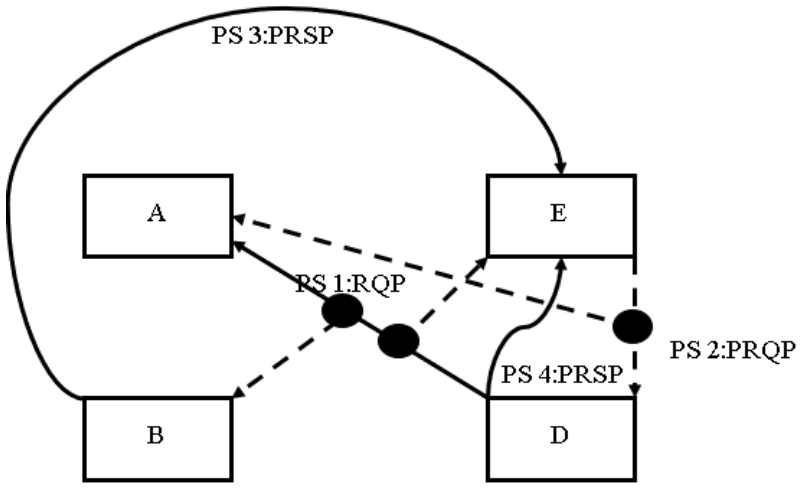


Рисунок 4.2 – Приклад підробленого запиту

Однак за умови атаки весь трафік, який A хоче надіслати до B, буде надіслано до D.

Тепер обговорюється використання активного зондування, щоб підкреслити, як воно створює різницю в послідовності ARP-пакетів у звичайних ситуаціях і ситуаціях атаки. Отримавши RQP від D до A (пакетна послідовність 1), диспетчерський контролер IDS в E надсилає запит на зонд ARP PRQP для запиту MAC-адреси, відповідної IP(B) (послідовність пакетів 2); PRQP – це запит ARP, надісланий від IDS, що має вихідний IP-MAC як IP(E)-MAC(E).

Оскільки зонд ARP є широкомовним, і B, і зловмисник D отримають зонд. B безумовно відповість на зонд (PRSP) IP(B)-MAC(B) до E (пакетна послідовність 3), оскільки передбачається, що не входить хост (B) завжди буде реагувати на зонд; це неконтрольована подія. PRSP – це пакет відповіді ARP, надісланий до IDS, тобто IP-MAC призначення – IP(E)-MAC(E).

Таким чином, програмно-технічний засіб виявлення вторгнень в IT-інфраструктури може знати, що запит, зроблений у послідовності пакетів 1, є

хибним (оскільки IDS мав IP(B)-MAC(D)), і може генерувати тривогу (а також відстежувати MAC-адрес зловмисника (D)).

Щоб уникнути самоідентифікації, зловмиснику D бажано відповідати на всі запити (PRSP) щодо MAC-адреси IP(B), за допомогою IP(B)-MAC(D); це неконтрольована подія, оскільки поведінку зловмисника не можна вважати відомою. Підводячи підсумок, прийде принаймні одна відповідь на запит (скажімо, надісланий для перевірки IP(B) і матиме справжній IP-MAC B. У разі підробки надійде більше однієї відповіді, які можуть мати різні MAC-адреси адреси.

У цьому прикладі передбачається, що справжній хост B відповідає (послідовність пакетів 3) перед зловмисником D (послідовність пакетів 4) на зонд ARP від E. Можна зазначити, що в двох відповідях різні MAC-адреси пов’язані з IP-адреса, яка досліджується, тобто коли MAC-адреса B асоціюється з IP-адресою B, а потім MAC-адреса D асоціюється з IP-адресою B. Таблиця 4.1перераховує послідовність ARP-пакетів для прикладу .

Таблиця 4.1 – Послідовність пакетів і подій у прикладі підробки запитів

PS:Events	SRCIP	SRCMAC	DestIP	DestMAC
PS1:RQP	IP(B)	MAC(D)	IP(A)	–
PS2:PRQP	IP(E)	MAC(E)	IP(B)	–
PS3:PRSP	IP(B)	MAC(B)	IP(E)	MAC(E)
PS4:PRSP	IP(B)	MAC(D)	IP(E)	MAC(E)

Послідовність ARP-трафіку, якщо пакет запиту (послідовність 1) є справжнім (тобто, IP(B)-MAC(B) надсилається до A від B), задається наступним чином. Для ARP-зонда, надісланого E для перевірки IP(B)-MAC(B) (пакетна послідовність 2), лише B відповість на E IP(B)-MAC(B) (пакетна послідовність 3). До часу T_{req} IDS отримає лише один PRSP (на відміну від двох у разі атаки). IDS вважає, що відповідь є справжньою (через лише одну PRSP).

4.3Дослідження швидкості, точностізасобами запропонованого програмно-технічного засобу виявлення вторгнень в IT-інфраструктури

Показниками для визначення ефективності IDS є точність, швидкість виявлення та споживання ресурсів.

У випадку пасивного споживання ресурсів програмно-технічним засобом виявлення вторгнень в IT-інфраструктури визначається за допомогою пам’яті та використання CPUсистемою, яка виконує IDS.

З іншого боку, у разі активного використання, оскільки через зондування генеруються додаткові повідомлення, використання пропускної здатності є ще одним показником споживання ресурсів.

IT-інфраструктура для проведення експериментальних досліджень Випробувальний стенд був створений для експериментального аналізу запропонованої IDS.

IT-інфраструктура складається з 6 комп’ютерних систем під керуванням операційних систем – Windows 10 (як робочі станції), openSUSELeap 15.3 (як маршрутизатор),openSUSELeap 15.3 (як робочі станції), КС з Kali 2022.1 є комп’ютерною системою-зловмисником, а такожокремакомп’ютерна системапід керуваннямopenSUSELeap 15.3 налаштована як IDS.

Ці комп’ютерні системи підключаються за допомогою комутатора CISCO Catalyst серії 3560 G з увімкненим дзеркальним відображенням портів для програмно-технічного засобу виявлення вторгнень в IT-інфраструктури.

IDS реалізовано на C++. Для кожного запиту ARP або пакету відповіді, отриманого в дзеркальному порту, наглядовий контролер генерує подію, який, у свою чергу, створює екземпляр детектора у стані s_1 і додається до активного списку. Після надходження наступної події (тобто RQP, PRQP, PRSP1, PRSP2) вона передається всім екземплярам детекторів у активному списку до відповідного поточного стану.

Кожен з них може здійснити перехід на ці події, якщо це можливо.

Якщо в будь-якому (нетермінальному) стані перехід до наступного стану неможливий, оголошується атака, а екземпляр детектора видаляється з активного списку (таким чином звільняється пам’ять).

Також видаляються всі екземпляри детекторів, які досягають термінального стану.

Іншими словами, для виявлення підробки ARP створюється багато екземплярів машини-детектора [49].

Кожна машина перевіряє, чи належить трасування, створене послідовністю подій ARP, мові, створеній діагностикою.

За допомогою розгортання інструментів генерації атак Ettercap, CainandAbel на хості D і спробується кілька сценаріїв підробки MAC-адрес.

Різна кількість пакетів атаки (до 1000) в секунду вводяться в тестовий стенд.

Таблиця 4.2ілюструє швидкість виявлення та точність у залежності від відсотка пакетів ARP, для яких надсилається запит.

Таблиця 4.2 - Статистика підробки ARP

% залучених пакетів	Кількість атак	Рівень виявлення (%)
100	1000	97.22
90	1000	85.36
80	1000	83.12
70	1000	77.55
60	1000	73.77
50	1000	69.33
40	1000	61.34
30	1000	59.69
20	1000	57.09
10	1000	51.23

Очевидно, що зі зменшенням кількості посилених зондових пакетів існує компроміс у швидкості виявлення.

Зниження швидкості виявлення зі зменшенням частоти зондування відбувається тому, що визначення підробленої пари IP-MAC (якої немає в таблиці Spoofed) можна здійснити лише шляхом відправки проби та отримання відповіді з невідповідною MAC-адресою.

Однак, що цікаво, швидкість зниження частоти виявлення нижча, ніж швидкість зниження рівня зондування.

Зокрема, із лише 10% зондованих пакетів рівень виявлення все ще перевищує 50%.

Причина в тому, що протягом певного періоду часу створюються таблиці Authenticated і Spoofed, де зберігаються всі справжні та підроблені пари IP-MAC. Ці таблиці можуть виявляти атаки за допомогою історії і не вимагають надсилання зондів; це відповідає події DTD.

Можна відзначити, що навіть для 100% зондування рівень виявлення не 100%, тобто є деякі помилкові негативи.

Це пов’язано з такими випадками, як підробка зловмисників тощо, які неможливо виявити.

Серед різних створених сценаріїв атаки 5% – це ті, де зловмисник обманює себе.

Це число навмисно тримається на низькому рівні, оскільки сценарій, коли зловмисник фальсифікує себе, не має жодного впливу.

Точність становить 100% (тобто помилкові спрацьовування рівні 0) у всіх випадках, оскільки підробка ARP виявляється, коли (i) пара IP-MAC, що перевіряється, знайдена в підробленій таблиці або (ii) для зонда приходить відповідь з не - відповідність MAC-адреси.

Випадок (i) містить пари IP-MAC, які вже виявлено як підроблені, і випадок (ii) ніколи не може статися за нормальних умов.

Таким чином, схема ніколи не надсилає сигнал тривоги для нормального випадку, в результаті точність становить 100%.

На рисунку 4.3показано використання пропускної спроможності через трафік ARP при різних відсотках перевірок. Оскільки зонди генерують додатковий трафік ARP (через сам зонд і відповіді), пропускна здатність зменшується зі зменшенням відсотка зондування.

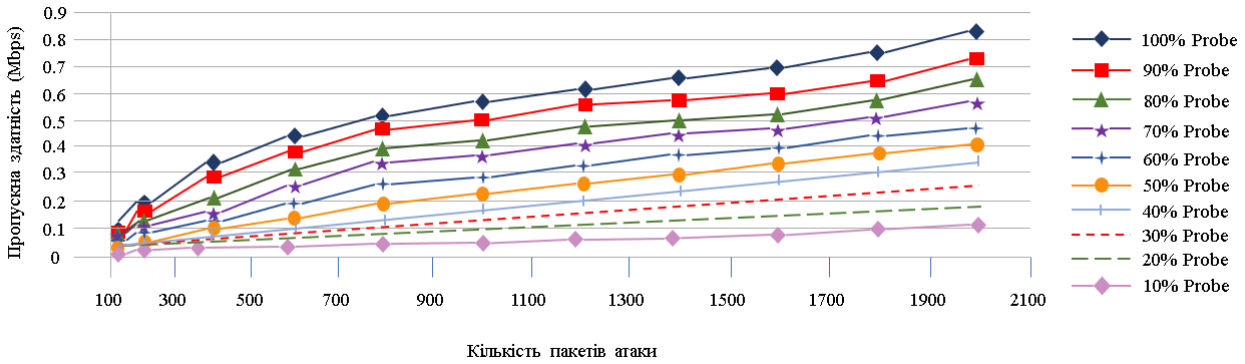


Рисунок 4.3 – Використання смуги пропускання при різному відсотку зондування

На рисунках 4.4і 4.5(червона лінія) показано використання центрального процесору і пам’яті (при 100% тестуванні) відповідно під час запуску IDS на хостіIntelCore i7 з 16 ГБ оперативної пам’яті. Завантаження центрального процесору і пам'яті падають із зменшенням відсотка тестування; оскільки тенденції подібні до пропускної здатності, не представляємо їх явно.

Рисунок 4.4і рисунок 4.5також порівнюють використання центрального процесору і пам'яті запропонованого IDS з показником [50]. Оскільки може бути створено багато екземплярів детектора, використання пам’яті та центрального процесору хостом, що виконує IDS, є допустимим і не перевантажує хост. Цей факт проілюстровано на рисунках 4.4.і 4.5.

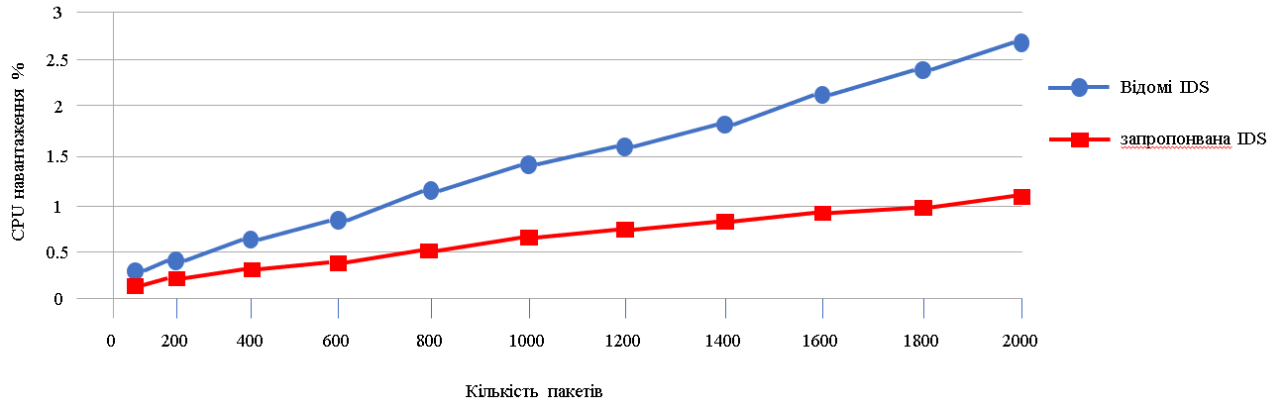


Рисунок 4.4 – Порівняння використання центрального процесору з IDS [50]

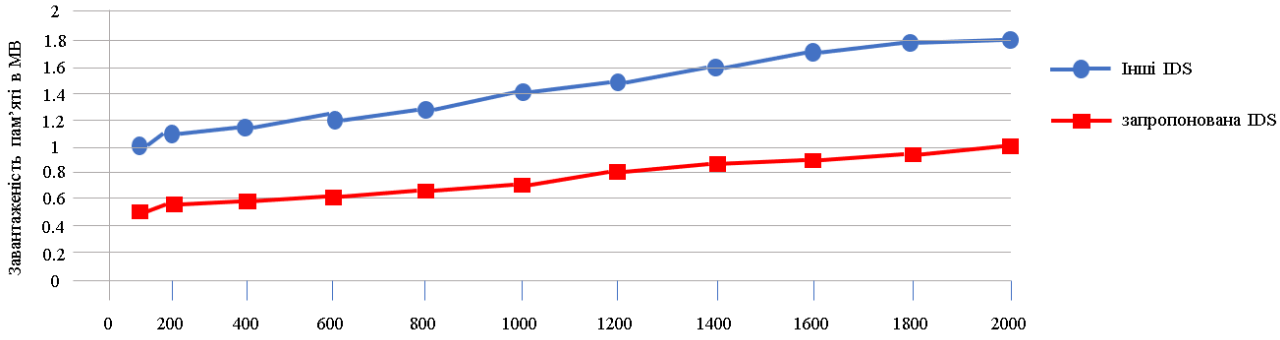


Рисунок 4.5 – Порівняння використання пам'яті з IDS [50]

Використання пропускної здатності як [50], так і запропонованого IDS є подібним. Це пов’язано з тим, що обидва IDS активні, тим самим надсилаючи запити для пар IP-MAC, справжність яких невідома.

Для кожного зонда є одна або дві реакції на основі нападу або нормального стану. Таким чином, додатковий трафік ARP подібний для обох IDS, тобто однакові вимоги до пропускної здатності.

Що стосується ресурсних витрат запропонованої схеми, можна відзначити, що використання центрального процесору, використання пам’яті та пропускної здатності, навіть при 100% зондуванні, мінімальні порівняно з ресурсами, доступними в сучасній інфраструктурі.

IT інфраструктура, створена для експериментів, показана на рисунку 3.1. IT інфраструктура складається з чотирьох КС (A-D) у підмережі 1 та ще однієї КС в підмережі 2, на яких працюють різні операційні системи. На КС A-D

працюють такі ОС: Windows 10, openSUSELeap 15.3, KaliLinux 2022.1 і KaliLinux 2022.1 відповідно.

КС D з KaliLinux 2022.1 діє як RC зловмисника, а машина C налаштована як IDS. Ці RC підключені до локальної мережі за допомогою комутатора CISCO catalyst серії 3560 G з увімкненим дзеркальним відображенням портів для системи C.

Підмережа 2 містить КС Z, на якій працює операційній системіopenSUSELeap 15.3.

Використовуємо чотири маршрутизатори R1, R2, R3 і R4 для створення мережі.

SNMP і NetFlow увімкнені в маршрутизаторі R1, який діє як шлюз для підмережі 1. Як маршрутизатори використовуються КCLinux з двома або трьома мережевими картами.

Встановлено пакет програмного забезпечення для маршрутизації Quagga [51] разом із пакетом Net-SNMP [52] для підтримки SNMP.

Демон zebraquagga обробляє основні функції маршрутизації. Крім того, на цих машинах увімкнено IP-пересилання, щоб можна було діяти як маршрутизатори.

Таблиці створюються в базі даних MySql. IDS розроблено з використанням мови C++. IDS має два випереджувальних модуля, а саме, перехоплювач пакетів і інжектор пакетів. Захоплення пакетів перехоплює пакети з мережі, фільтрує пакети ICMP і викликає відповідний алгоритм залежно від типу пакета.

Генератор пакетів генерує різні зонди, необхідні для перевірки повідомлень ICMP. Інструменти генерації атак Ettercap, Sing, Hping2 та інші коди атак на C були розгорнуті на машині D, і було апробовано кілька сценаріїв підробки повідомлень ICMP.

Таблиця 4.3 показує швидкість виявлення залежно від відсотка пакетів, які перевіряються для виявлення атак на основі ICMP NHU.

При 100% зондуванні рівень виявлення близький до 98%. Решта 2% випадків не виявлено через перевантаження мережі.

При 20% зондування рівень виявлення падає приблизно до 37%.

Точність у всіх випадках становить 100%, а швидкість виявлення пропорційна компромісу при зондуванні.

Таблиця 4.3 – Статистика виявлення атак на основі ICMP NHU

%пакетів	Кількість атак	Рівень виявлення вторгень
100	160951	97.87
80	160951	88.57
60	160951	71.32
40	160951	58.21
20	160951	37.15

На рисунках 4.5 і 4.6 3.12 показано кількість додаткового трафіку ICMP, створеного через зонди, надіслані для перевірки повідомлень ICMP.

На графіку на рисунку 4.5 показано обсяг трафіку за нормальної роботи за наявності та відсутності IDS.

Помічено, що додатковий трафік, який створюється внаслідок зондування, є незначним.

На графіку 4.6 3.12 показано трафік, який генерується, коли IDS запущено і в мережі відбуваються атаки на основі повідомлень NHU.

У цьому випадку також можна спостерігати невелику кількість додаткового трафіку, який генерується IDS через зонди.

Враховуючи невеликий відсоток ICMP-трафіку в мережевому трафіку, істотного збільшення трафіку не спостерігається.

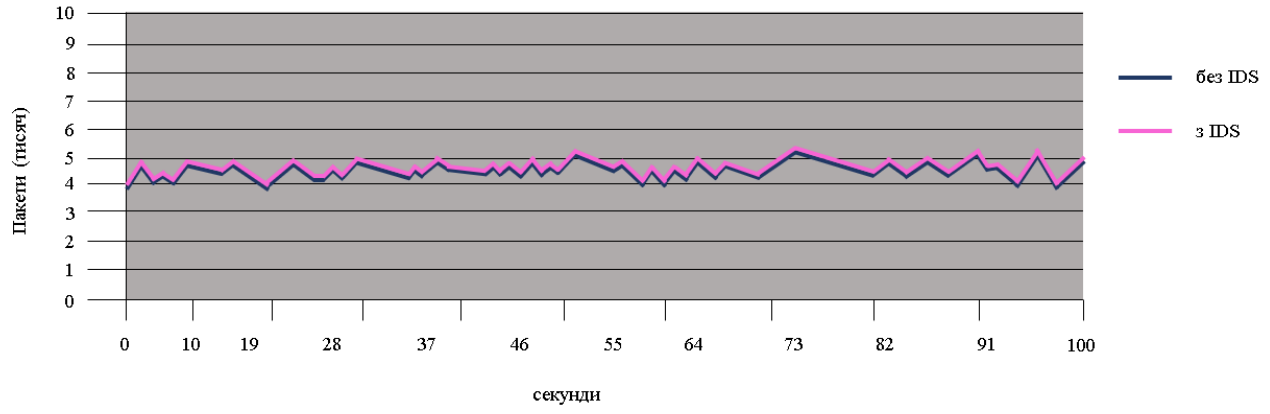


Рисунок 4.5 – Трафік у ICMP NHU Нормальний стан

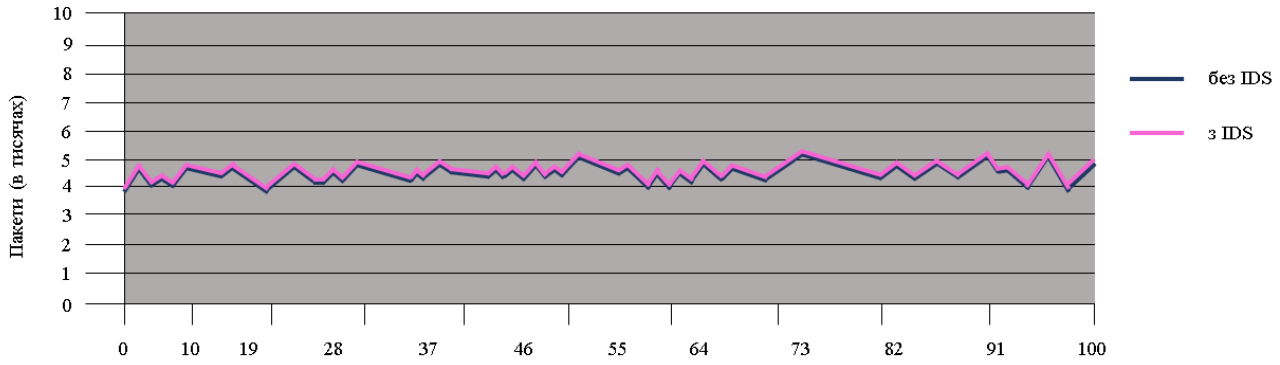


Рисунок 4.6 – Трафік в умовах атаки на основі ICMP

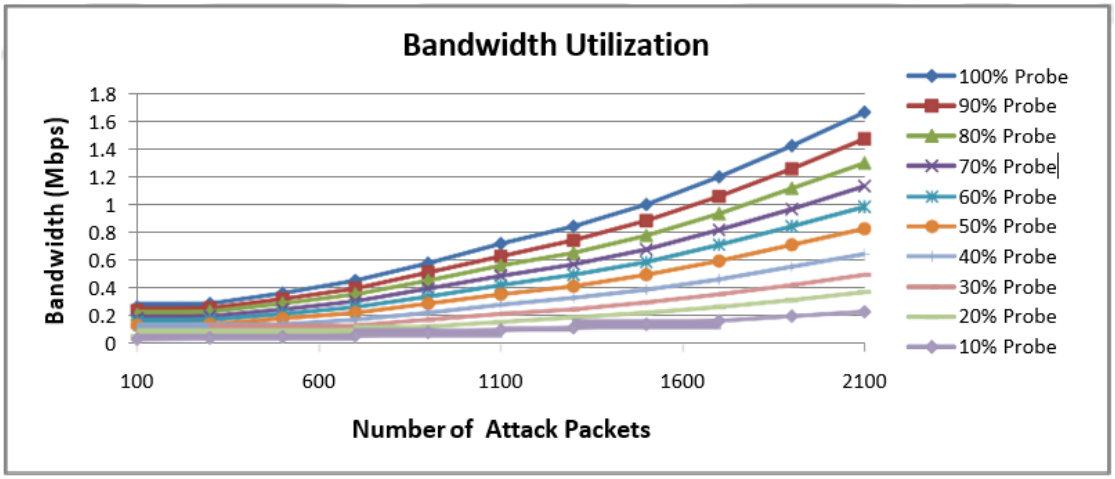


Рисунок 4.7 – Використання пропускної здатності IDS

На тестовому стенді впроваджено до 10 000 пакетів атаки протягом 100 секунд і виміряли використання процесора, на якому працює IDS, і використання пам’яті хоста, що виконує IDS, і використання пропускної здатності. На рисунках 4.7, 4.8, 4.9 показано використання ЦП, пам’яті та пропускної спроможності відповідно під час запуску IDS на машині IntelCore i7 (з OpenSUSELeap 15.3) при різному обсязі тестування.

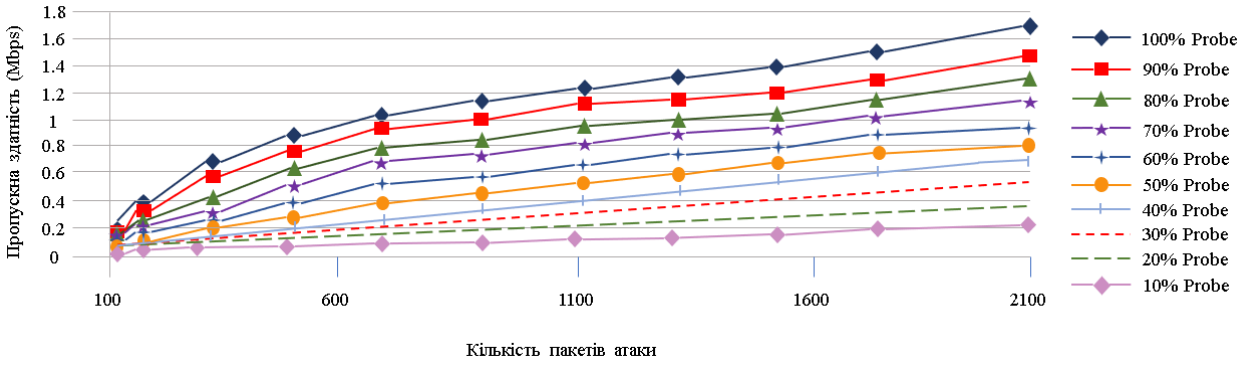


Рисунок 4.8 – Використання центрального процесору IDS

Можна відзначити, що в найгіршому випадку виконаних експериментальних досліджень (тобто максимальна кількість введених пакетів атаки) використання ЦП становить близько 3%, загальне використання пам’яті системою становить близько 1 Гб, а пропускна здатність – близько 2 Мбіт/с. Отже, можна зробити висновок, що при великій кількості пакетів атаки також IDS працює без проблем у ресурсному плані.

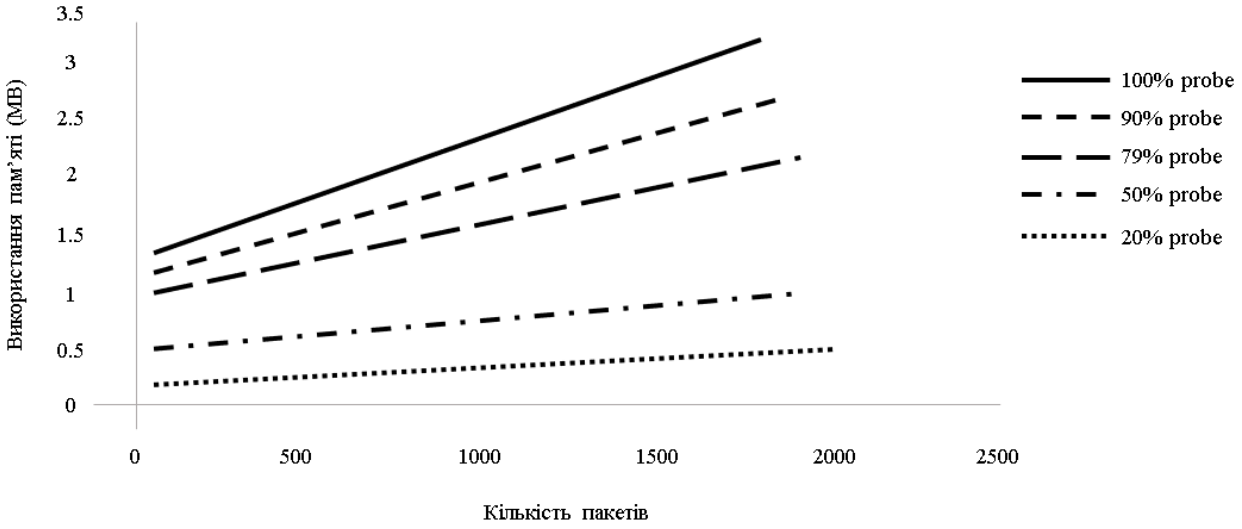


Рисунок 4.9 – Використання пам'яті IDS

4.4 Висновок

В результаті реалізації удосконаленого методу було реалізовано програмно-технічний засіб виявлення вторгнень в IT-інфраструктури.

Програмно-технічний засіб виявлення вторгнень в IT-інфраструктури

використовує DES на основі I-діагностики. У схемі використовується активний механізм для виявлення атак без використання дорогого криптографічного механізму. Також схема не вимагає зміни в стеку протоколів і не порушує архітектуру шарів.

Запропонований IDS дозволяє виявити атаку шляхом діагностики лише на тих шляхах, де за несправністю слідує індикаторна подія. Це дозволяє виявляти атаки ICMP, незважаючи на такі проблеми, як перевантаження мережі. Проблема вибуху стану в структурі I-Diagnosability вирішується шляхом доповнення структури змінними моделі. Зменшений I-детектор (так званий RI-детектор) був розроблений шляхом усунення зайвих станів, і було показано, що здатність виявлення атак не порушена.

Атаки ICMP можна виявити, розділивши звичайну послідовність подій атаки, шляхом діагностики певних шляхів за допомогою I-діагностики DES. Але у випадку певних атак, як-от LowRate TCP DoS-атаки, розділення звичайних подій і подій атаки неможливе за допомогою I-діагностики або активної DES-фреймворку, оскільки атака та справжні сліди можуть не відрізнятися чітко, але можуть відрізнятися з певною ймовірністю.

Практичне застосування реалізованого програмно-технічного засобу виявлення вторгнень в IT-інфраструктури продемонструвало підвищення достовірності та ефективності точності виявлення вторгнень в IT-інфраструктури до 97%.

ВИСНОВКИ

Магістерська робота присвячена розв’язанню науково-практичної задачі підвищення достовірності та ефективності точності виявлення вторгнень в IT-інфраструктури.

В першому розділі було досліджено системи виявлення вторгнень. Також було проаналізовано таксономію IDS на основі виявлення атак, зокрема IDS на основі підпису та IDS на основі аномалій. було проаналізовано обмеження існуючих IDS.

В розділі також досліджено основні аспекти функціонування вторгнень засобами атаки на основі ARP, атаки через повідомлення ICMP, засобами повільних DoS-атак, а також атак на основі NDP.

Виявлено недоліки відомих методів програмно-технічних засобів систем виявлення вторгнень в IT-інфраструктури.

Зроблено висновок та постановку задачі щодо необхідності удосконалення методу виявлення вторгнень в IT-інфраструктури.

В другому розділі запропоновано моделювання процесу виявлення вторгнень в IT-інфраструктури. З цією метою було застосовано апарат дискретно-подійне моделювання (DES). Вказаний математичний апарат дозволив здійснити наступні дослідження, а саме виконати моделювання обробки підроблених пакетів при атаці на IT-інфраструктуру, побудувати модель атак на IT-інфраструктуруз вимірюваністю., побудувати модель атак на IT-інфраструктуру з керованістю, побудувати моделювання відмов, побудувати DES-модель атаки типу спуфінгу ARP.

В третьому розділі було запропоновано удосконалений метод виявлення вторгнень в IT-інфраструктури, що ґрунтується на застосуванні DES-моделей, які характеризуються дискретним простором станів і певною динамікою, керованою подіями в нормальних умовах, а також за кожного з умов атаки. В основі методу лежить механізм оцінки стану (детектор), який спостерігає за послідовністю подій, створених системою, щоб вирішити, чи відповідають стани, через які система проходить, нормальній чи несправній моделі DES, і

далі застосовується механізм активного зондування.Також в розділі було представлено, наприклад, виявлення вторгнення атакою NHU. Метод містить наступні кроки: моніторинг мережі, застосування механізму активного зондування, перевірка правильності пакетів, обробка підроблених пакетів, моделювання DES в нормальних умовах та в умовах здійснення вторгнення, застосування детектора.

В четвертому розділі представлено реалізацію удосконаленого методу було реалізовано програмно-технічний засіб виявлення вторгнень в IT-інфраструктури, який використовує DES на основі I-діагностики. Запропонований IDS дозволяє виявити атаки шляхом діагностики лише на тих шляхах, де за несправністю слідує індикаторна подія. Практичне застосування реалізованого програмно-технічного засобу виявлення вторгнень в IT-інфраструктури продемонструвало підвищення достовірності та ефективності точності виявлення вторгнень в IT-інфраструктури до 97%.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ERT statistics. URL: http://www.cert.org/stats. (дата звернення: 25.04.2022).

2. M. G. Goudaand C. T. Huang. CSI/FBI computercrimeandsecuritysurvey. *ComputerSecurityJournal*. XV(2). 2020

3. R. Heady, G. Lugar, M. Servilla, and A. Maccabe. Thearchitectureof a networklevelintrusiondetectionsystem. Technicalreport.*DepartmentofComputerScience.*UniversityofNewMexico. 2013.

4. M. Roesch. SNORT - lightweightintrusiondetectionfornetworks. *USENIX SystemAdministrationConference*. 2015.pages 229–238.

5. P. A. Porrasand P. G. Neumann. EMERALD: eventmonitoringenablingresponsestoanomalouslivedisturbances. *National Information SystemsSecurityConference*. 2016.1.1.1. pages 1–13.

6. V. Chandol, A. Banerjee, and V. Kumar. Anomalydetection: A survey. *ACM ComputingSurvey*. 2009.1.1.2. Vol. 41(3). pp. 1–58.

7. A. Patchaand J. Park. Anoverviewofanomalydetectiontechniques: Existingsolutionsandlatesttechnologicaltrends. *ComputerNetworks*. 2007. 1.1.2. Vol. 51(12). pp. 3448–3470.

8. C. M. Kozierok. TCP/IP Guide. 1 edition. October 2015.

9. Лисенко Сергій, Сокальський Дмитро, Михасько Яна. Дослідження методів виявлення кібератак в комп'ютерних мережах як засобів до побудови резильєнтних до вторгнень ІТ-інфраструктур. *ComputerSystemsand Information Technologies*. 2021. № 3. с.31-35

10. Сокальський Д. О., Лисенко С. М.Програмно-технічний засіб виявлення вторгнень в ІТ-Інфраструктури. *ЗбірникнауковихпрацьXIIIВсеукраїнськоїнауково-практичноїконференції «Актуальніпроблемикомп'ютерних наук АПКН-2021».*Хмельницький, 2021. с. 228-229.

11. C. L. Abadand R. I. Bonilla. Ananalysisontheschemesfordetectingandpreventing ARP cachepoisoningattacks.

InternationalConferenceonDistributedComputingSystemsWorkshops. 2017.pages 60–67.

12. D. Fadhilahand M. I. Marzuki. PerformanceAnalysisof IDS Snortand IDS SuricatawithMany-CoreProcessorinVirtualMachinesAgainstDos/DDoS Attacks, *2nd InternationalConferenceonBroadbandCommunications, WirelessSensorsandPowering (BCWSP)*, 2020, pp. 157-162, doi: 10.1109/BCWSP50066.2020.9249449.

13. H. Hendrawan, P. Sukarnoand M. A. Nugroho. QualityofService (QoS) ComparisonAnalysisofSnort IDS andBro IDS ApplicationinSoftwareDefineNetwork (SDN) Architecture,*7th InternationalConferenceon Information andCommunicationTechnology (ICoICT),* 2019, pp. 1-7, doi: 10.1109/ICoICT.2019.8835211.

14. N. Chalaemwongwanand W. Kurutach. A PracticalNationalDigital ID FrameworkonBlockchain (NIDBC). *15th InternationalConferenceonElectricalEngineering/Electronics, Computer, Telecommunicationsand Information Technology (ECTI-CON)*, 2018, pp. 497-500, doi: 10.1109/ECTICon.2018.8620003.

15. C. P. XuanQui, D. HongQuang, P. T. Duy, D. ThiThuHienand V. -H. Pham, Strengthening IDS againstEvasionAttackswith GAN-basedAdversarialSamplesin SDN-enablednetwork.*RIVF InternationalConferenceonComputingandCommunication Technologies (RIVF),* 2021, pp. 1-6, doi: 10.1109/RIVF51545.2021.9642111.

16. V. Ramachandranand S. Nandi. Detecting ARP spoofing: Anactivetechnique. *InternationalConferenceon Information SystemsSecurity*. 2015.pages 239–250.

17. Z. Trabelsiand K. Shuaib. Maninthemiddleintrusiondetection.*GlobalTelecommunicationsConference 2006*. 2016.pages 1–6.

18. D. Thorsleyand D. Teneketzis. Intrusiondetectionincontrolleddiscreteeventsystems. *ConferenceonDecisionandControl*. 2016.pages 6047–6054.

19. N. Hubbali, S. Biswas, S. Roopa, R. Ratti, and S. Nandi. LAN attackdetectionusingdiscreteeventsystems. *ISA Transactions*. 2010. Vol. 50(1). pp. 119–130.

20. N. Hubbali, S. Biswas, S. Roopa, R. Ratti, S .Nandi, F.A. Barbhuiya, A. Sur, and V. Ramachandran. A DES approachtointrusiondetectionsystemfor ARP spoofingattacks. *MediterraneanConferenceonControlandAutomation*. 2011.pages 695–700.

21. C. G. Cassandrasand S. Lafortune. IntroductiontoDiscreteEventSystems. KluwerAcademicPublishers. 1 edition. 2015.

22. M. Sampath, S. Lafortune, and D. Teneketzis. Activediagnosisofdiscrete-eventsystems. *IEEE TransactionsonAutomaticControl*. 2014. Vol. 43.pp. 908–929.

23. H. Lee, V. Thing, Y. Xu, M. Ma. ICMP TracebackwithCumulativePath, anEfficientSolutionfor IP Traceback . *LNCS, Information andCommunicationsSecurity*. 2013. Vol. 2836. pp. 124– 135.

24. J. Zhang, J. Liu, Z. Xu, J. Li, X. Ye . TRDP : a TrustedRouterDiscoveryProtocol.*InternationalSymposiumonCommunicationsand Information Technologies ISCIT '07*. 2017. pages 660 – 665.

25. A Kuzmanovicand E. W. Knightly. Low-rate TCP-targeteddenialofserviceattacks: theshrewvs. themiceandelephants. *ConferenceonApplications, Technologies, Architectures, andProtocolsforComputerCommunications*. 2003.1.2.3,4.2, 4.2. pages 75–86.

26. R. Mahajan, S. Floyd, and D. Wetherall. Controllinghigh-bandwidthflowsatthecongestedrouter. *InternationalConferenceonNetworkProtocols*. 2011. pages 192–20.

27. A. Kuzmanovicand E. W. Knightly. Low-rate TCP-targeteddenialofserviceattacksandcounterstrategies. *IEEE/ACM TransactionsonNetworking*. 2016. Vol. 14(4). pp. 683–696.

28. H. Sun, J. C. S. Lui, and D. K. Y. Yau. Defendingagainstlow-rate TCP attacks: Dynamicdetectionandprotection. *InternationalConferenceonNetworkProtocol*. 2014.pages 196–205.

29. A. Shevtekar, K. Anantharam, and N. Ansari. Lowrate TCP denial-of-serviceattackdetectionatedgerouters. *IEEE CommunicationsLetters*. 2015. Vol. 9(4). pp. 363–365.

30. Y. Kwok, R. Tripathi, Y. Chen, and K. Hwang. Hawk: haltinganomalieswithweightedchokingtorescuewell-behaved TCP sessionsfromshrewddosattacks. *InternationalConferenceonNetworkingandMobileComputing*. 2015.pages 423–432.

31. XiapuLuoandRocky K. C. Chang. On a new class ofpulsingdenial-of-serviceattacksandthedefense. *NetworkandDistributedSystemSecuritySymposium*. 2015.pages 61–79.

32. C. Chang, S. Lee, B. Lin, and J. Wang. Thetamingoftheshrew: mitigatinglow-rate TCP-targetedattack. *IEEE TransactionsonNetworkandServiceManagement*. 2010.1.2.37. Vol. (1). pp. 1–13.

33. D. Thorsleyand D. Teneketzis. DiagnosabilityofStochasticDiscrete-EventSystems.

34. *IEEE TransactiononAutomaticControl.*April 2005.1.2.3,3.6,4.1,4.2,4.3, 4.6. Vol. 50(4). pp. 76–492.

35. S. Thomson, T. Narten, and T. Jinmei. IPv6 statelessaddressautoconfiguration. RFC 4862.*InternetEngineeringTaskForce*.September 2017.

36. Ed. J. Arkko, J. Kempf, B. Zill, and P. Nikander. SEcureNeighborDiscovery (SEND).

37. Ed. J. Arkko, J. Kempf, B. Zill, and P. Nikander. CryptographicallyGeneratedAddresses (CGA). RFC 3972. *InternetEngineeringTaskForce.*March 2015.

38. S. Jiangand R. Kumar. FailureDiagnosisofDiscrete-EventSystemswithLinear-TimeTemporalLogicspecifications. *IEEE TransactionsonAutomaticControl*. 2014.Vol. 49(6). pp. 934 – 945.

39. M. Izadi, M. Bonsangue, and D. Clarke. Buchiautomataformodelingcomponentconnectors. *Software&SystemsModeling*.2011. Vol. 10(2). pp. 183–200.

40. SeanWhalen. AnIntroductionto ARP Spoofing. *Technicalreport*.April 2013.

41. Han-WeiHsiao, Cathy S. Lin, andSsu-YangChang. Constructinganarpattackdetectionsystemwith SNMPtrafficdatamining.*InternationalConferenceonElectronicCommerce.*2019.pages 341–345.

42. Mohamed G. GoudaandChin-TserHuang. A secureAddressResolutionProtocol. *ComputerNetworks*. 2017. Vol. 41(1).pp. 57–71.

43. WesamLootah, WilliamEnck, andPatrickMcDaniel. TARP: Ticket-basedaddressresolutionprotocol. *AnnualComputerSecurityApplicationsConference.* 2015.pages 106–116.

44. F. Gont. ICMP attacksagainst TCP. RFC 5927. *InternetEngineeringTaskForce*.2016.

45. Cimatti, E.M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebas- tiani, and A. Tacchella. Nusmv 2: Anopensourcetoolforsymbolicmodelchecking. *InternationalConferenceonComputerAidedVerification*. 2002.1.2.4,5.1, 5.4.3.3. pages 359–64.

46. T. Narten, E. Nordmark, and W. Simpson. Securityfeaturesin IPv6. Whitepaper. *SANS Institute*. 2012.

47. H. Rafiee, A Alsa'deh, and C Meinel. Winsend: Windows secureneighbordiscovery. *InternationalConferenceonSecurityof Information andNetworks*.2011.pages 243–246.

48. G. Bansal, N. Kumar, F. A.Barbhuiya, S. Biswas, S. Nandi. ScalableImplementationofActiveDetectionMechanismfor LAN BasedAttacks.*InternationalConferenceonNetworkSecurity&Applications (CNSA 2016).*Chennai.(CCIS). pp 258 – 267.

49. N. Hubballi, S. Biswas, F.A.Barbhuiya,Roopa S., R. Ratti, S. Nandi.DiscreteEventSystemApproachtoIntrusionDetectionSystemfor ARP Attacks.*MediterraneanConferenceonControland Automation-2010 (MED 2013)*. Marrakech. Morocco. (IEEE).pp 695 – 700.

50. M. M. Shurman, R. M. Khraisand A. A. Yateem. IoTDenial-of-ServiceAttackDetectionandPreventionUsingHybrid IDS. *InternationalArabConferenceon Information Technology (ACIT),* 2019, pp. 252-254, doi: 10.1109/ACIT47987.2019.8991097.

51. QuaggaRoutingSoftwareSuite. URL: https://www.quagga.net/ (датазвернення: 15.11.2021).

52. SimpleNetworkManagementProtocol (SNMP). URL: http://www.net-snmp.org/ (дата звернення: 15.11.2021).

Додаток А

(обов’язковий)

Код систеного программного забезпечення програмно-технічний засіб виявлення вторгнень в IT-інфраструктури

```
#include <tbb/spin_mutex.h>
#include <stdio.h>
#include <sstream>

#include "CountersBuffer.h"

using namespace std;
CountersBuffer::CountersBuffer() {
}

CountersBuffer::CountersBuffer(const CountersBuffer&orig) {
}

CountersBuffer::~CountersBuffer() {
}

void CountersBuffer::AddAll(int tcpAcc, int udpAcc, int icmpAcc, int tcpAckAcc,
int tcpFinAcc, int tcpSynAcc, int totalAcc) {
lock.acquire(CountersMutex);
    this->tcpAcc += tcpAcc;
    this->udpAcc += udpAcc;
    this->icmpAcc += icmpAcc;
    this->tcpAckAcc += tcpAckAcc;
    this->tcpFinAcc += tcpFinAcc;
    this->tcpSynAcc += tcpSynAcc;
    this->totalAcc += totalAcc;
    this->grandTotal += totalAcc;
lock.release();
}

void CountersBuffer::AddTotalAcc() {
lock.acquire(CountersMutex);
    this->totalAcc++;
lock.release();
}
```

```
void CountersBuffer::AddTcpSynAcc() {
lock.acquire(CountersMutex);
    this->tcpSynAcc++;
lock.release();
}

void CountersBuffer::AddTcpFinAcc() {
lock.acquire(CountersMutex);
    this->tcpFinAcc++;
lock.release();
}

void CountersBuffer::AddTcpAckAcc() {
lock.acquire(CountersMutex);
    this->tcpAckAcc++;
lock.release();
}

void CountersBuffer::AddIcmpAcc() {
lock.acquire(CountersMutex);
    this->icmpAcc++;
lock.release();
}

void CountersBuffer::AddUdpAcc() {
lock.acquire(CountersMutex);
    this->udpAcc++;
lock.release();
}

void CountersBuffer::AddTcpAcc() {
lock.acquire(CountersMutex);
    this->tcpAcc++;
lock.release();
}

void CountersBuffer::ZeroCounters() {
lock.acquire(CountersMutex);
    this->tcpAcc = 0;
    this->udpAcc = 0;
    this->icmpAcc = 0;
    this->tcpAckAcc = 0;
```

```
this->tcpFinAcc = 0;
    this->tcpSynAcc = 0;
this->totalAcc = 0;
lock.release();
}

void CountersBuffer::PrintValues(){

cout<<tcpAcc<<"|"<<udpAcc<<"|"<<icmpAcc<<"|"<<tcpAckAcc<<"|"<<tcpFinAcc<<"|"<<tcpSynAcc
<<"|"<<totalAcc<<std::endl;
}

void CountersBuffer::PrintAndRestartGrandTotal(){
cout<<"Grand Total "<<grandTotal<<endl;
grandTotal=0;
}

#ifndef COUNTERSBUFFER_H
#define      COUNTERSBUFFER_H
#include <tbb/mutex.h>
#include <iostream>
class CountersBuffer {
public:
CountersBuffer();
CountersBuffer(const CountersBuffer&orig);
    virtual ~CountersBuffer();
    void AddTotalAcc();
    void AddTcpSynAcc();
    void AddTcpFinAcc();
    void AddTcpAckAcc();
    void AddIcmpAcc();
void AddUdpAcc();
    void AddTcpAcc();
void AddAll(int tcpAcc, int udpAcc, int icmpAcc, int tcpAckAcc, int tcpFinAcc,
int tcpSynAcc, int totalAcc);
    void ZeroCounters();
    void PrintValues();
    void PrintAndRestartGrandTotal();
private:
    int tcpAcc, udpAcc, icmpAcc, tcpAckAcc, tcpFinAcc, tcpSynAcc, totalAcc;
    long int grandTotal;
    typedef tbb::spin_mutexCountersMutexType;
CountersMutexTypeCountersMutex;
```

```
CountersMutexType::scoped_lock lock;
};

#endif /* COUNTERSBUFFER_H */


#include "DBData.h"
DBData::DBData(int numActiveFilters) {
    counters = new int[numActiveFilters];
}

DBData::DBData(const DBData&orig) {
}

DBData::~DBData() {
}

void DBData::SetCounters(int* counters) {
    this->counters = counters;
}

int* DBData::GetCounters() const {
    return counters;
}

void DBData::SetTsp(std::time_t tsp) {
    this->tsp = tsp;
}

std::time_tDBData::GetTsp() const {
return tsp;
}

#ifndef DBDATA_H
#define      DBDATA_H
#include<ctime>

class DBData {
public:
DBData(int numActiveFilters);
DBData(const DBData&orig);
    virtual ~DBData();
    void SetCounters(int* counters);
```

```
    int* GetCounters() const;
    void SetTsp(std::time_t tsp);
    std::time_tGetTsp() const;
private:
    std::time_t tsp;
    int* counters;
};

#endif /* DBDATA_H */
#include "FilterStatus.h"

FilterStatus::FilterStatus() {
tcp = true;
    //TODO restore this filter
udp = true;
icmp = true;
tcpSYN = true;
tcpFIN = true;
tcpACK = true;
numberOfActiveFilters = 0;
}

FilterStatus::FilterStatus(const FilterStatus&orig) {
}

FilterStatus::~FilterStatus() {
}

int FilterStatus::getNumberOfActivatedFilters() {
    return numberOfActiveFilters;
}

bool FilterStatus::isTCPActivated() {
    return tcp;
}

bool FilterStatus::isUDPActivated() {
    return udp;
}

bool FilterStatus::isICMPActivated() {
    return icmp;
}
```

```
bool FilterStatus::isTCPSYNActivated() {
    return tcpSYN;
}

bool FilterStatus::isTCPFINActivated() {
    return tcpFIN;
}

bool FilterStatus::isTCPACKActivated() {
    return tcpACK;
}



void FilterStatus::activateTCP() {
    if (!tcp) {
tcp = true;
numberOfActiveFilters++;
    }
}

void FilterStatus::deactivateTCP() {
    if (tcp) {
tcp = false;
numberOfActiveFilters--;
    }
}

void FilterStatus::activateUDP() {
    if (!udp) {
udp = true;
numberOfActiveFilters++;
    }
}

void FilterStatus::deactivateUDP() {
    if (udp) {
udp = false;
numberOfActiveFilters--;
    }
}

void FilterStatus::activateICMP() {
```

```
    if (!icmp) {
icmp = true;
numberOfActiveFilters++;
    }
}

void FilterStatus::deactivateICMP() {
    if (icmp) {
icmp = false;
numberOfActiveFilters--;
    }
}

void FilterStatus::activateTCPSYN() {
    if (!tcpSYN) {
tcpSYN = true;
numberOfActiveFilters++;
    }
}

void FilterStatus::deactivateTCPSYN() {
    if (tcpSYN) {
tcpSYN = false;
numberOfActiveFilters--;
    }
}

void FilterStatus::activateTCPFIN() {
    if (!tcpFIN) {
tcpFIN = true;
numberOfActiveFilters++;
    }
}

void FilterStatus::deactivateTCPFIN() {
    if (tcpFIN) {
tcpFIN = false;
numberOfActiveFilters--;
    }
}

void FilterStatus::activateTCPACK() {
    if (!tcpACK) {
```

```
tcpACK = true;
numberOfActiveFilters++;
    }
}

void FilterStatus::deactivateTCPACK() {
    if (tcpACK) {
tcpACK = false;
numberOfActiveFilters--;
    }
}

#ifndef FILTERSTATUS_H
#define      FILTERSTATUS_H

class FilterStatus {
public:
FilterStatus();
FilterStatus(const FilterStatus&orig);
    virtual ~FilterStatus();

    bool isTCPActivated();
    bool isUDPActivated();
    bool isICMPActivated();
    bool isTCPSYNActivated();
    bool isTCPFINActivated();
    bool isTCPACKActivated();
    int getNumberOfActivatedFilters();
    //Manipulation functions
    void activateTCP();
    void activateUDP();
    void activateICMP();
    void activateTCPACK();
    void activateTCPFIN();
    void activateTCPSYN();

    void deactivateTCP();
    void deactivateUDP();
    void deactivateICMP();
    void deactivateTCPACK();
    void deactivateTCPFIN();
    void deactivateTCPSYN();
private:
```

```
    bool tcp;
    bool udp;
    bool icmp;
    bool tcpACK;
    bool tcpFIN;
    bool tcpSYN;
    int numberOfActiveFilters;
};

#endif /* FILTERSTATUS_H */

#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <cstring>
//---
#include "NetPcap.h"
#include "PacketAnalyserDispatcher.h"
using namespace std;

//XML Scanner
//DataManager
//ThreadDispatcher
int istatus;
int inicId = -1;

void runCapture();
void shutdown();
void restartAll();
void setNic();

PacketsBufferpacketsBuffer;
NetPcapnetPcap(&packetsBuffer);
PacketAnalyserDispatcher dispatcher(&packetsBuffer);

int main(int nArg, char* pszArgs[]) {
    //Initializing objects
    //Parsing XML properties
    //si no hay argumentospreguntarporunanic
    if (nArg<= 1) {
printf("Asking for a nic\n");
setNic();
    } else {
```

```
printf("Looks like the nic was preconfigured %d\n", atoi(pszArgs[1]));
inicId = atoi(pszArgs[1]);
        //netPcap.setNetworkDevice(inicId);
        char* nicName="lo";
netPcap.setNetworkDevice(inicId, nicName);//Use allways lo
    }

    //If there is an error with the nics
    if (inicId == -2) {
        return -1;
    }

    //Starting and forking threads
    //Simple cmd input
    while (istatus != 2) {
runCapture();
printf("> ");
cin>>istatus;
        if (istatus == 0) {
            //Si yaestacorrientoimprimir que yaestacorriendo
            //runCapture();
        } else if (istatus == 1) {
restartAll();
        } else if (istatus< 0 &&istatus> 2) {
printf("Wrong command\n");
        }
    }

    //If threads are alive execute shutdown
    shutdown();
cout<< "bye!";
}

void setNic() {
    //print nics
    while (inicId == -1) {
netPcap.chooseNetworkDevice(inicId);
    }
}

void runCapture() {
printf("Running analysis\n");
netPcap.openNetworkDevice();
```

```
    //netPcap.run();
netPcap.start();
dispatcher.start();
}

void shutdown() {
printf("Shutting down\n");
dispatcher.countersBuffer.PrintAndRestartGrandTotal();
netPcap.close();
dispatcher.close();
}

void restartAll() {
printf("restarting\n");
    shutdown();
}

#include <iostream>

#include "NetPcap.h"
#include "FilterStatus.h"
#include "XMLProperties.h"

using namespace std;

NetPcap::NetPcap() {
    //this->packagesCount=0;
}

NetPcap::NetPcap(PacketsBuffer *buffer) {
    this->packetsBuffer = buffer;
}

NetPcap::NetPcap(const NetPcap&orig) {
}

NetPcap::~NetPcap() {
}

void NetPcap::chooseNetworkDevice(int &inicId) {
    char *dev;
pcap_if_t *alldevs;
```

```
pcap_findalldevs(&alldevs, errbuf);
    int i = 0;

    for (device = alldevs; device; device = device->next) {
printf("%d. %s", i++, device->name);
        if (device->description)
printf(" (%s)\n", device->description);
        else
printf(" (No description available)\n");
    }
    if (i> 0) {
printf("Select nic> ");
cin>>inicId;
        if (inicId>i || inicId<-2) {
printf("Wrong nic number\n");
inicId = -1;
        }
        /* Jump to the selected adapter */
        for (device = alldevs, i = 0; i<inicId; device = device->next, i++);
printf("Selected %s device\n", device->name);
    } else {
printf("No NICs available\n");
inicId = -2;
    }
}

//Work with numbers

void NetPcap::setNetworkDevice(int &inicId) {
    char *dev;
pcap_if_t *alldevs;

pcap_findalldevs(&alldevs, errbuf);
    int i = 0;
    for (device = alldevs; device; device = device->next) {
printf("%d. %s", i++, device->name);
        if (device->description)
printf(" (%s)\n", device->description);
        else
printf(" (No description available)\n");
    }
    /* Jump to the selected adapter */
    if (i> 0) {
```

```
        for (device = alldevs, i = 0; i<inicId; device = device->next, i++);
printf("Selected %d - %s device\n", i, device->name);
    } else {
printf("No NICs available\n");
inicId = -2;
    }
}

//Work with chars

void NetPcap::setNetworkDevice(int &inicId, char* snicName) {
    char *dev;
pcap_if_t *alldevs;

pcap_findalldevs(&alldevs, errbuf);
    int i = 0;
    for (device = alldevs; device; device = device->next) {
printf("%d. %s \n", i++, device->name);
        if (strcmp(device->name, snicName) == 0) {
            break;
        }
    }

    if (i> 0) {
printf("Selected %d - %s device\n", i, device->name);
    } else {
printf("No NICs available\n");
inicId = -2;
    }
}

void NetPcap::openNetworkDevice() {
    if ((pcap = pcap_open_live(device->name, SNAP_LEN, 1, 20, errbuf)) == NULL) {
fprintf(stderr, "\nError opening adapter\n");
        return;
    }
    //Creating filters
    string expressionF = buildExpression();
    char expresion[expressionF.size()];
strcpy(expresion, expressionF.c_str());
    /* Compile and apply the filter */
cout<< "Expresion: " <<expresion<<endl;
    if (pcap_compile(pcap, &filter, expresion, 0, net) == -1) {
```

```
        fprintf(stderr, "Couldn't parse filter %s: %s\n", expresion, pcap_geterr(pcap));
                return;
            }
            if (pcap_setfilter(pcap, &filter) == -1) {
        fprintf(stderr,    "Couldn't    install    filter    %s:    %s\n",    expresion,
pcap_geterr(pcap));
                return;
            }
        }

        /*Wrapper function from C to C++*/
        extern NetPcapnetPcap;
        void count_packet(u_char *args, const struct pcap_pkthdr *header, const u_char
*packet) {
            /*pcappacketcappacket = {packet, header};
        packetsBuffer.addPacket(cappacket);
            */

        netPcap.count_packet_handler(header, packet);
        }

        void NetPcap::count_packet_handler(const struct pcap_pkthdr *header, const u_char
*packet) {
            //cout<< "Metodollamadodesdeelobjeto\n";
        pcappacketcappacket = {packet, header};
        packetsBuffer->addPacket(cappacket);
            /*packagesCount++;
        printf("Packet number %d:\n", packagesCount);*/
        }

        void NetPcap::run() {
            //pcap_loop(pcap, -1, NetPcap::count_packet, NULL);
        cout<<"Capturing packages\n";
        pcap_loop(pcap, -1, count_packet, NULL);

        }

        /**
         * Worker method
         **/
        void NetPcap::start() {
        analyzerThread = boost::thread(&NetPcap::run, this);
        }
```

```
void NetPcap::join() {
analyzerThread.join();
analyzerThread.detach();
cout<<"Detached thread\n";
}

void NetPcap::interrupt() {
analyzerThread.interrupt();
}

void NetPcap::close() {
pcap_breakloop(pcap);
pcap_close(pcap);
    interrupt();
    join();
}

string NetPcap::buildExpression() {
    string filter = "";
    list<string>separatedFilters;
FilterStatus fs;
    if (fs.isTCPActivated()) {
separatedFilters.push_back("tcp");
    }
    if (fs.isUDPActivated()) {
separatedFilters.push_back("udp");
    }
    if (fs.isICMPActivated()) {
separatedFilters.push_back("icmp");
    }
    if (fs.isTCPACKActivated()) {
separatedFilters.push_back("(tcp[tcpflags] & (tcp-ack) != 0)");
    }
    if (fs.isTCPFINActivated()) {
separatedFilters.push_back("(tcp[tcpflags] & (tcp-fin) != 0)");
    }
    if (fs.isTCPSYNActivated()) {
separatedFilters.push_back("(tcp[tcpflags] & (tcp-syn) != 0)");
    }

    for (int i = 0; i<separatedFilters.size(); i++) {
        filter += separatedFilters.front();
```

```
            if ((i + 1) != separatedFilters.size()) {
                filter += " or ";
            }
        separatedFilters.pop_front();
        }
        //filter = "port " + xmlProps.GetPortToFilter() + " and (" + filter + ")";

        if (xmlProps.GetPortToFilter() != -1) {
            string portFilter = "port " + boost::lexical_cast<std::string >
(xmlProps.GetPortToFilter());
            filter = portFilter + " and (" + filter + ")";
        }
        return filter;
    }

    void NetPcap::got_packet(u_char *args, const struct pcap_pkthdr *header, const
u_char *packet) {
        static int count = 1; /* packet counter */

        /* declare pointers to packet headers */
        const struct sniff_ethernet *ethernet; /* The ethernet header [1] */
        const struct sniff_ip *ip; /* The IP header */
        const struct sniff_tcp *tcp; /* The TCP header */
        const char *payload; /* Packet payload */

        int size_ip;
        int size_tcp;
        int size_payload;

    printf("\nPacket number %d:\n", count);
        count++;

        /* define ethernet header */
        ethernet = (struct sniff_ethernet*) (packet);

        /* define/compute ip header offset */
    ip = (struct sniff_ip*) (packet + SIZE_ETHERNET);
    size_ip = IP_HL(ip)*4;
        if (size_ip< 20) {
    printf("   * Invalid IP header length: %u bytes\n", size_ip);
            return;
        }
```

```
    /* print source and destination IP addresses */
printf("       From: %s\n", inet_ntoa(ip->ip_src));
printf("         To: %s\n", inet_ntoa(ip->ip_dst));

    /* determine protocol */
    switch (ip->ip_p) {
        case IPPROTO_TCP:
printf("   Protocol: TCP\n");
            break;
        case IPPROTO_UDP:
printf("   Protocol: UDP\n");
            return;
        case IPPROTO_ICMP:
printf("   Protocol: ICMP\n");
            return;
        case IPPROTO_IP:
printf("   Protocol: IP\n");
            return;
        default:
printf("   Protocol: unknown\n");
            return;
    }

    /*
     *  OK, this packet is TCP.
     */

    /* define/compute tcp header offset */
tcp = (struct sniff_tcp*) (packet + SIZE_ETHERNET + size_ip);
size_tcp = TH_OFF(tcp)*4;
    if (size_tcp< 20) {
printf("   * Invalid TCP header length: %u bytes\n", size_tcp);
        return;
    }

printf("   Src port: %d\n", ntohs(tcp->th_sport));
printf("   Dst port: %d\n", ntohs(tcp->th_dport));

    return;
};

void NetPcap::print_hex_ascii_line(const u_char *payload, int len, int offset) {
    int i;
```

```
    int gap;
    const u_char *ch;

    /* offset */
printf("%05d   ", offset);

    /* hex */
ch = payload;
    for (i = 0; i<len; i++) {
printf("%02x ", *ch);
ch++;
        /* print extra space after 8th byte for visual aid */
        if (i == 7)
printf(" ");
    }
    /* print space to handle line less than 8 bytes */
    if (len< 8)
printf(" ");

    /* fill hex gap with spaces if not full line */
    if (len< 16) {
        gap = 16 - len;
        for (i = 0; i< gap; i++) {
printf("   ");
        }
    }
printf("   ");

    /* ascii (if printable) */
ch = payload;
    for (i = 0; i<len; i++) {
        if (isprint(*ch))
printf("%c", *ch);
        else
printf(".");
ch++;
    }

printf("\n");

    return;
};
```

```
void NetPcap::print_payload(const u_char *payload, int len) {
    int len_rem = len;
    int line_width = 16; /* number of bytes per line */
    int line_len;
    int offset = 0; /* zero-based offset counter */
    const u_char *ch = payload;

    if (len<= 0)
        return;

    /* data fits on one line */
    if (len<= line_width) {
print_hex_ascii_line(ch, len, offset);
        return;
    }

    /* data spans multiple lines */
    for (;;) {
        /* compute current line length */
line_len = line_width % len_rem;
        /* print line */
print_hex_ascii_line(ch, line_len, offset);
        /* compute total remaining */
len_rem = len_rem - line_len;
        /* shift pointer to remaining bytes to print */
ch = ch + line_len;
        /* add offset */
        offset = offset + line_width;
        /* check if we have line width chars or less */
        if (len_rem<= line_width) {
            /* print last line and get out */
print_hex_ascii_line(ch, len_rem, offset);
            break;
        }
    }

    return;
};

#ifndef NETPCAP_H
#define      NETPCAP_H
#include <pcap.h>
#include <iostream>
```

```
#include <cstring>
#include <cstdlib>
#include <list>
#include <vector>
#include <ctype.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <boost/lexical_cast.hpp>
#include <boost/thread.hpp>

#include "XMLProperties.h"
#include "PacketsBuffer.h"
// <editor-fold defaultstate="collapsed" desc="struct definitions">
/* default snap length (maximum bytes per packet to capture) */
#define SNAP_LEN 1518
/* ethernet headers are always exactly 14 bytes [1] */
#define SIZE_ETHERNET 14
/* Ethernet addresses are 6 bytes */
#define ETHER_ADDR_LEN    6
/* Ethernet header */
struct sniff_ethernet {
u_charether_dhost[ETHER_ADDR_LEN];    /* destination host address */
u_charether_shost[ETHER_ADDR_LEN];    /* source host address */
u_shortether_type;                     /* IP? ARP? RARP? etc */
};

/* IP header */
struct sniff_ip {
u_charip_vhl;                 /* version << 4 | header length >> 2 */
u_charip_tos;                 /* type of service */
u_short ip_len;                 /* total length */
u_short ip_id;                  /* identification */
u_short ip_off;                 /* fragment offset field */
        #define IP_RF 0x8000            /* reserved fragment flag */
        #define IP_DF 0x4000            /* dont fragment flag */
        #define IP_MF 0x2000            /* more fragments flag */
        #define IP_OFFMASK 0x1fff       /* mask for fragmenting bits */
u_charip_ttl;                 /* time to live */
u_char  ip_p;                   /* protocol */
u_short ip_sum;                 /* checksum */
```

```
        struct  in_addrip_src,ip_dst;  /* source and dest address */
};
#define IP_HL(ip)               (((ip)->ip_vhl) & 0x0f)
#define IP_V(ip)                (((ip)->ip_vhl) >> 4)
typedef u_inttcp_seq;

struct sniff_tcp {
u_shortth_sport;               /* source port */
u_shortth_dport;               /* destination port */
tcp_seq th_seq;                 /* sequence number */
tcp_seq th_ack;                 /* acknowledgement number */
u_char  th_offx2;               /* data offset, rsvd */
        #define TH_OFF(th)      (((th)->th_offx2 & 0xf0) >> 4)
u_charth_flags;
        #define TH_FIN  0x01
        #define TH_SYN  0x02
        #define TH_RST  0x04
        #define TH_PUSH 0x08
        #define TH_ACK  0x10
        #define TH_URG  0x20
        #define TH_ECE  0x40
        #define TH_CWR  0x80
        #define                                                         TH_FLAGS
(TH_FIN|TH_SYN|TH_RST|TH_ACK|TH_URG|TH_ECE|TH_CWR)
u_short th_win;                 /* window */
u_shortth_sum;                 /* checksum */
u_short th_urp;                 /* urgent pointer */
};

//</editor-fold>


class NetPcap {
public:
NetPcap();
NetPcap(PacketsBuffer *buffer);
NetPcap(const NetPcap&orig);
    virtual ~NetPcap();
    void chooseNetworkDevice(int &inicId);
    void setNetworkDevice(int &inicId);
    void setNetworkDevice(int &inicId, char* snicName);
    void openNetworkDevice();
    void start();
```

```
            void run();
            void join();
            void close();
            void interrupt();
            std::string buildExpression();
            int packagesCount;
            void  count_packet_handler(const  struct  pcap_pkthdr  *header,  const  u_char
*packet);
        private:
        pcap_if_t *device;
        pcap_t *pcap;
            char errbuf[PCAP_ERRBUF_SIZE];
            struct bpf_program filter; /* The compiled filter */
            bpf_u_int32 net;
            //std::string buildExpression();
        XMLPropertiesxmlProps;
            void got_packet(u_char *args, const struct pcap_pkthdr *header, const u_char
*packet);
        PacketsBuffer *packetsBuffer;
            //void  count_packet(u_char  *args,  const  struct  pcap_pkthdr  *header,  const
u_char *packet);
            void print_payload(const u_char *payload, int len);
            void print_hex_ascii_line(const u_char *payload, int len, int offset);
            //
            boost::thread analyzerThread;

        };

        #endif /* NETPCAP_H */

        #include "PacketAnalyser.h"

        PacketAnalyser::PacketAnalyser() {
        }

        PacketAnalyser::PacketAnalyser(tbb::concurrent_vector<pcappacket>*  packetVector,
CountersBuffer* countersBuffer, int start, int lenght) {
            this->packetVector = packetVector;
            this->countersBuffer = countersBuffer;
            this->start = start;
            //printf("Constructing analyzer %d\n",lenght);
            this->lenght = lenght;
        }
```

```
PacketAnalyser::PacketAnalyser(const PacketAnalyser&orig) {
}

PacketAnalyser::~PacketAnalyser() {
}

tbb::task* PacketAnalyser::execute() {
    int mainThreads = 2;
    int threshold = 10000;

    if (lenght< threshold) {
        //printf("Computing directly");
computeDirectly();
    } else {
        int split = lenght / mainThreads;
tbb::task_list list;
        this->set_ref_count(3);
        //printf("Spliting Length %d\n", lenght);
PacketAnalyser&   a   =   *new(   allocate_child())   PacketAnalyser(packetVector,
countersBuffer, start, split);
        int seconSplit;
        if((lenght%2)==0){
seconSplit = split;
        }else{
seconSplit = split+1;
        }
PacketAnalyser&   b   =   *new(   allocate_child())   PacketAnalyser(packetVector,
countersBuffer, start + split, seconSplit);
list.push_back(a);
list.push_back(b);
        // Start a running and wait for all children (a and b).
spawn_and_wait_for_all(list);
    }
    return NULL;
}

//void PacketAnalyser::computeDirectly() {
//    int packNu=0;
//    int tcpAcc, udpAcc, icmpAcc, tcpAckAcc, tcpFinAcc, tcpSynAcc, totalAcc;
//    for (int i = start; (i< start + lenght) && (i<packetVector->size()); i++) {
//        //processPackage(packetVector->at(i));
//        packNu++;
```

```
	//		//processPackage(packetVector->at(i), tcpAcc, udpAcc, icmpAcc, tcpAcc,
tcpFinAcc, tcpSynAcc, totalAcc);
	//	}
	//	 printf("Computed %d - packnu %d - processed %d \n",lenght, packNu,
totalAcc);
	//}

	void PacketAnalyser::computeDirectly() {
	    int packNu = 0;
	    int tcpAcc = 0, udpAcc = 0, icmpAcc = 0, tcpAckAcc = 0, tcpFinAcc = 0,
tcpSynAcc = 0, totalAcc = 0;
	    for (int i = start; (i< start + lenght) && (i<packetVector->size()); i++) {
	pcappacketentirePacket = packetVector->at(i);
	packNu++;
		const struct pcap_pkthdr *header = entirePacket.header;
		const u_char *packet = entirePacket.packet;
		/* declare pointers to packet headers */
		const struct sniff_ethernet *ethernet; /* The ethernet header [1] */
		const struct sniff_ip *ip; /* The IP header */
		const struct sniff_tcp *tcp; /* The TCP header */
		const char *payload; /* Packet payload */

		int size_ip;
		int size_tcp;
		int size_payload;

		//printf("\nPacket number %d:\n", count);
		ethernet = (struct sniff_ethernet*) (packet);

		/* define/compute ip header offset */
	ip = (struct sniff_ip*) (packet + SIZE_ETHERNET);
	size_ip = IP_HL(ip)*4;
		if (size_ip< 20) {
	printf("   * Invalid IP header length: %u bytes\n", size_ip);
		    return;
		}

	totalAcc++;
		/* determine protocol */
		switch (ip->ip_p) {
		    case IPPROTO_TCP:
	tcpAcc++;
			//Procesarsubtipo
```

```
                    break;
                case IPPROTO_UDP:
                    //printf("   Protocol: UDP\n");
udpAcc++;
                    break;
                case IPPROTO_ICMP:
icmpAcc++;
                    //printf("   Protocol: ICMP\n");
                    break;
                default:
                    //printf("   Protocol: unknown\n");
                    break;
            }
            //processPackage(packetVector->at(i), tcpAcc, udpAcc, icmpAcc, tcpAcc,
tcpFinAcc, tcpSynAcc, totalAcc);
        }
        countersBuffer-
>AddAll(tcpAcc,udpAcc,icmpAcc,tcpAckAcc,tcpFinAcc,tcpSynAcc,totalAcc);
        //printf("Computed %d - packnu %d - processed %d \n", lenght, packNu,
totalAcc);
    }


    //void        PacketAnalyser::processPackage(pcappacket&entirePacket,        int
&tcpAcc,int&udpAcc,int&icmpAcc,int&tcpAckAcc,int&tcpFinAcc,int&tcpSynAcc,int&totalAcc)
{

    void PacketAnalyser::processPackage(pcappacket&entirePacket) {
        int tcpAcc, udpAcc, icmpAcc, tcpAckAcc, tcpFinAcc, tcpSynAcc, totalAcc;
        const struct pcap_pkthdr *header = entirePacket.header;
        const u_char *packet = entirePacket.packet;
        /* declare pointers to packet headers */
        const struct sniff_ethernet *ethernet; /* The ethernet header [1] */
        const struct sniff_ip *ip; /* The IP header */
        const struct sniff_tcp *tcp; /* The TCP header */
        const char *payload; /* Packet payload */

        int size_ip;
        int size_tcp;
        int size_payload;

        //printf("\nPacket number %d:\n", count);
        ethernet = (struct sniff_ethernet*) (packet);
```

```
        /* define/compute ip header offset */
    ip = (struct sniff_ip*) (packet + SIZE_ETHERNET);
    size_ip = IP_HL(ip)*4;
        if (size_ip< 20) {
    printf("   * Invalid IP header length: %u bytes\n", size_ip);
            return;
        }

    totalAcc++;
        /* determine protocol */
        switch (ip->ip_p) {
            case IPPROTO_TCP:
    tcpAcc++;
                //Procesarsubtipo
                break;
            case IPPROTO_UDP:
                //printf("   Protocol: UDP\n");
    udpAcc++;
                break;
            case IPPROTO_ICMP:
    icmpAcc++;
                //printf("   Protocol: ICMP\n");
                break;
            default:
                //printf("   Protocol: unknown\n");
                break;
        }

        /* print source and destination IP addresses */

        //printf("tcpAcc: %d, udpAcc: %d, icmpAcc: %d \n",tcpAcc, udpAcc, icmpAcc);
        //countersBuffer-
>AddAll(tcpAcc,udpAcc,icmpAcc,tcpAckAcc,tcpFinAcc,tcpSynAcc,totalAcc);
    }

    #ifndef PACKETANALYSER_H
    #define      PACKETANALYSER_H
    #include "NetPcap.h"
    #include "CountersBuffer.h"
    #include <tbb/task.h>
    #include <tbb/task_scheduler_init.h>
    #include <iostream>
```

```
class PacketAnalyser: public tbb::task {
public:
PacketAnalyser();
PacketAnalyser(tbb::concurrent_vector<pcappacket>* packetVector, CountersBuffer*
countersBuffer, int start, int lenght);
PacketAnalyser(const PacketAnalyser&orig);
    virtual ~PacketAnalyser();

private:
tbb::task* execute();
    void computeDirectly();
tbb::concurrent_vector<pcappacket>* packetVector;
    //void               processPackage(pcappacket&entirePacket,               int
&tcpAcc,int&udpAcc,int&icmpAcc,int&tcpAckAcc,int&tcpFinAcc,int&tcpSynAcc,int&totalAcc);
    void processPackage(pcappacket&entirePacket);
    int start;
    int lenght;
CountersBuffer* countersBuffer;
};

#endif /* PACKETANALYSER_H */

#include "PacketAnalyserDispatcher.h"

using namespace std;

PacketAnalyserDispatcher::PacketAnalyserDispatcher() {
}

PacketAnalyserDispatcher::PacketAnalyserDispatcher(PacketsBuffer *buffer) {
    this->packetsBuffer = buffer;
}

PacketAnalyserDispatcher::PacketAnalyserDispatcher(const
PacketAnalyserDispatcher&orig) {
}

PacketAnalyserDispatcher::~PacketAnalyserDispatcher() {
}

void PacketAnalyserDispatcher::start() {
dispatcherThread = boost::thread(&PacketAnalyserDispatcher::run, this);
}
```

```
	void PacketAnalyserDispatcher::run() {
	    /*boost::posix_time::seconds workTime(5);
	cout<< "Worker: running" <<endl;
	    boost::this_thread::sleep(workTime);*/
	    while (true) {
	        try{
	runAnalysis();
	            boost::posix_time::seconds workTime(2);
	            boost::this_thread::sleep(workTime);
	        }catch(boost::thread_interrupted const&){
	            break;
	        }
	    }

	}

	void PacketAnalyserDispatcher::runAnalysis() {
	tbb::concurrent_vector<pcappacket>analysisVector = packetsBuffer-
>getAnalisysVector();
	    int copiedSize = analysisVector.size();
	    //cout<< "Vector size " <<copiedSize<<endl;
	    //cout<< "Before cleaning" <<packetsBuffer->getSize() <<endl;
	    //boost::posix_time::seconds workTime(1);
	    //boost::this_thread::sleep(workTime);
	    //tbb::task_scheduler_initinit(8);
	PacketAnalyser& a = *new(tbb::task::allocate_root())
PacketAnalyser(&analysisVector, &countersBuffer, 0, copiedSize);
	tbb::task::spawn_root_and_wait(a);

	packetsBuffer->cleanHeadElements(copiedSize);
	    //cout<< "After cleaning" <<packetsBuffer->getSize() <<endl;
	    //countersBuffer.PrintValues();
	    //persist data
	countersBuffer.ZeroCounters();
	}

	void PacketAnalyserDispatcher::join() {
	dispatcherThread.join();
	dispatcherThread.detach();
	cout<< "Detached thread\n";
	}
```

```
void PacketAnalyserDispatcher::interrupt() {
dispatcherThread.interrupt();
}

void PacketAnalyserDispatcher::close() {
    interrupt();
    join();

#ifndef PACKETANALYSERDISPATCHER_H
#define      PACKETANALYSERDISPATCHER_H


#include "NetPcap.h"
#include "PacketAnalyser.h"
#include "CountersBuffer.h"

class PacketAnalyserDispatcher {
public:
PacketAnalyserDispatcher();
PacketAnalyserDispatcher(PacketsBuffer *buffer);
PacketAnalyserDispatcher(const PacketAnalyserDispatcher&orig);
    virtual ~PacketAnalyserDispatcher();
    void start();
    void run();
    void runAnalysis();
    void join();
    void close();
    void interrupt();
CountersBuffercountersBuffer;
private:
    boost::thread dispatcherThread;
PacketsBuffer *packetsBuffer;

//std::vector<pcappacket>toSimpleVector(tbb::concurrent_queue<pcappacket>&queue);
};

#endif /* PACKETANALYSERDISPATCHER_H */

#include "PacketsBuffer.h"
using namespace std;

PacketsBuffer::PacketsBuffer() {
}
```

```
PacketsBuffer::PacketsBuffer(const PacketsBuffer&orig) {
}

PacketsBuffer::~PacketsBuffer() {
}

void PacketsBuffer::addPacket(pcappacket packet){
packetsList.push(packet);
    //cout<<packetCount++ <<endl;
}

void PacketsBuffer::cleanHeadElements(int size){
pcappacket temp;
    for (int i = 0; i< size; i++) {
packetsList.try_pop(temp);
    }
}

int PacketsBuffer::getSize(){
packetsList.unsafe_size();
};

tbb::concurrent_queue<pcappacket>PacketsBuffer::getAnalisysList(){
    //copy
    return tbb::concurrent_queue<pcappacket>(packetsList);
};

tbb::concurrent_vector<pcappacket>PacketsBuffer::getAnalisysVector(){
    //copy
tbb::concurrent_queue<pcappacket>copyQueue(packetsList);
    return PacketsBuffer::toVector(copyQueue);
};

tbb::concurrent_vector<pcappacket>PacketsBuffer::toVector(tbb::concurrent_queue<p
cappacket>&queue) {
    //copy
tbb::concurrent_vector<pcappacket>returnVector;
    //cout<< "Copying queue size " <<queue.unsafe_size() <<endl;
pcappacket packet;
    int i=0;
    while (queue.try_pop(packet)) {
returnVector.push_back(packet);
```

```
        }
        return returnVector;
    };

    #ifndef PACKETSBUFFER_H
    #define      PACKETSBUFFER_H

    #include<tbb/concurrent_queue.h>
    #include<tbb/concurrent_vector.h>
    #include <iostream>
    #include <cstring>
    #include <cstdlib>
    struct pcappacket {
        const u_char *packet;
        const struct pcap_pkthdr *header;
    };

    class PacketsBuffer {
    public:
    PacketsBuffer();
    PacketsBuffer(const PacketsBuffer&orig);
        virtual ~PacketsBuffer();
        void addPacket(pcappacket packet);

    tbb::concurrent_vector<pcappacket>getAnalisysVector();
        void cleanHeadElements(int size);
        int getSize();
    private:
        //TODO volverestatico
    tbb::concurrent_queue<pcappacket>packetsList;
        int packetCount;
    tbb::concurrent_queue<pcappacket>getAnalisysList();
    tbb::concurrent_vector<pcappacket>toVector(tbb::concurrent_queue<pcappacket>&queu
e);
    };

    #endif /* PACKETSBUFFER_H */

    #include "XMLProperties.h"
    using namespace std;

    XMLProperties::XMLProperties() {
    portToFilter = 8999;
```

```
}

XMLProperties::XMLProperties(const XMLProperties&orig) {
}

XMLProperties::~XMLProperties() {
}

#ifndef XMLPROPERTIES_H
#define      XMLPROPERTIES_H

#include <iostream>
#include <cstring>
#include <cstdlib>
class XMLProperties {
public:
XMLProperties();
XMLProperties(const XMLProperties&orig);
    virtual ~XMLProperties();

    int GetSocketsPort() const {
        return socketsPort;
    }

    bool IsSocketsOn() const {
        return socketsOn;
    }

    std::string GetDbArchiveTableName() const {
        return dbArchiveTableName;
    }

    bool IsDbArchiveOn() const {
        return dbArchiveOn;
    }

    std::string GetDbFatName() const {
        return dbFatName;
    }

    bool IsDbFatOn() const {
        return dbFatOn;
    }
```

```
    std::string GetDbPassword() const {
        return dbPassword;
    }

    std::string GetDbUsername() const {
        return dbUsername;
    }

    std::string GetDbLocation() const {
        return dbLocation;
    }

    std::string GetDbPort() const {
        return dbPort;
    }

    std::string GetDbHostName() const {
        return dbHostName;
    }

    std::string GetDbName() const {
        return dbName;
    }

    int GetPortToFilter() const {
        return portToFilter;
    }

    int GetAmmountOfIntervals() const {
        return ammountOfIntervals;
    }

    int GetIntervalOfAnalysis() const {
        return intervalOfAnalysis;
    }


private:
    int intervalOfAnalysis;
    int ammountOfIntervals;
    int portToFilter;
```

```
    // Database related properties
    std::string dbName;
    std::string dbHostName;
    std::string dbPort;
    std::string dbLocation;
    std::string dbUsername;
    std::string dbPassword;
    bool dbFatOn;
    std::string dbFatName; // Fast Access Table = FAT
    bool dbArchiveOn;
    std::string dbArchiveTableName;

    // Socket related
    bool socketsOn;
    int socketsPort;
};

#endif /* XMLPROPERTIES_H */
```

ДодатокБ

(обов'язковий)

Копіяпублікації

INTERNATIONAL SCIENTIFIC JOURNAL
«COMPUTER SYSTEMS AND INFORMATION TECHNOLOGIES»

UDC 004.93
DOI: 10.31891/CSIT-2021-5-4

SERGII LYSENKO, DMYTRO SOKALSKYI, IANA MYKHASKO
Khmelnytskyi National University, Khmelnytskyi, Ukraine

METHODS FOR CYBERATTACKS DETECTION IN THE COMPUTER NETWORKS AS A MEAN OF RESILIENT IT-INFRASTRUCTURE CONSTRUCTION: STATE-OF-ART

The paper presents a state-of-art of the methods for cyberattacks detection in the computer networks. The main accent was made on the concept of the resilience for the IT infrastructure. The concept of cyber resilience in the terms of cybersecurity was presented. The survey includes the set of approaches devoted to the problem of construction resilient infrastructures. All investigated approaches are aimed to construct and maintain infrastructure's resilience for cyberattacks resistance. Mentioned techniques and frameworks keep the main principles to assure resilience. To do this there exists some requirements to construct such infrastructure: IT infrastructure has to include the set ready to use measures of preparation concerning the possible cyber threats; it must include the set of special measures for the protection, as well as for cyberattacks detection; important issue and required is the possibility to respond the attack and to be able to absorb the negative attacks' impact; IT infrastructure must be as adaptive as it is possible, because today the dynamic of the attacks mutation is very high; IT infrastructure must be recoverable after the attacks were performed. In addition, the state-of-art found out that known approaches have domain-specific usage and it is important to develop new approaches and frameworks for the cyberattacks detection in the computer networks as a means of resilient IT-infrastructure construction.

Keywords: cyberattack, IT infrastructure, malware, computer systems, resilience, detection efficiency, network traffic

СЕРГІЙ ЛИСЕНКО, ДМИТРО СОКАЛЬСЬКИЙ, ЯНА МИХАСЬКО
Хмельницький національний університет

ДОСЛІДЖЕННЯ МЕТОДІВ ВИЯВЛЕННЯ КІБЕРАТАК В КОМП'ЮТЕРНИХ МЕРЕЖАХ ЯК ЗАСОБІВ ДО ПОБУДОВИ РЕЗИЛЬЄНТНИХ ДО ВТОРГНЕНЬ IT-ІНФРАСТРУКТУР

IT-інфраструктура стає все більш взаємопов'язаною, тоді як кіберзагрози для критичної інфраструктури стають все більш складними, від яких важко захиститися. Кібербезпека акцентує увагу на створенні засобів захисту, щоб запобігти втраті конфіденційності, цілісності та доступності цифрової інформації та систем, але останніми роками кібератаки продемонстрували, що жодна система не є непроникною. Кібер резильєнтність стала додатковим пріоритетом, який спрямований на те, щоб IT-інфраструктури могли підтримувати суттєві рівні продуктивності, навіть якщо можливості погіршуються внаслідок кібератаки. У статті представлено сучасні методи виявлення кібератак у комп'ютерних мережах. Основний акцент був зроблений на концепції стійкості для IT-інфраструктури. Було представлено поняття кіберстійкості з точки зору кібербезпеки. Дослідження включає комплекс підходів, присвячених проблемі побудови резильєнтних інфраструктур. Усі досліджувані підходи спрямовані на створення та підтримку резильєнтності інфраструктури до кібератак. Згадані методи та фреймворки зберігають основні принципи забезпечення резильєнтності. Для цього існують певні вимоги до побудови такої інфраструктури: IT-інфраструктура повинна включати набір готових до використання заходів підготовки щодо можливих кіберзагроз; має включати комплекс спеціальних заходів щодо захисту, а також виявлення кібератак; важливою і необхідною проблемою є можливість реагувати на атаку та мати можливість поглинати вплив негативних атак; IT-інфраструктура має бути максимально адаптивною, тому що сьогодні динаміка мутації атак дуже висока; IT-інфраструктура повинна бути відновлена після здійснення атак. Крім того, сучасний дослідження показало, що відомі підходи мають нішеве застосування і важливо розробляти нові підходи та засоби для виявлення кібератак у комп'ютерних мережах як засобу побудови резильєнтних IT-інфраструктур.

Ключові слова: шкідливе програмне забезпечення, живучість, комп'ютерні системи, достовірність виявлення, кібератака, мережний трафік.

Introduction

IT infrastructures have deeply into all aspects of our life. The cyber-physical systems based on the different infrastructures are a new concept today and are the next generation of system that must secure and be ready to resist the cyberthreats today [1-9]. In recent years, network cyberattacks on the IT infrastructure of information systems is still growing [10-14]. Thus, in June 2010, the security experts have found a botnet attack "StuxNet", which threated the industrial system, infected with computer systems and networks around the world [15].

Since the botnet malware StuxNet have appeared, a great variety of new cyber threads IT infrastructure had appeared [16-17]. Furthermore, new attacks involve new approaches and system information, that makes it possible to overcome the protection of IT infrastructure. That's why there exists an important task to investigate, maintain, and develop new methods for cyberattacks detection in the computer networks as a mean of resilient IT-infrastructure construction.

This research is a state-of-art of the methods for cyberattacks detection in the computer networks, that present the set of approach devoted to the problem of construction of resilient IT-infrastructures.

Concept of cyber resilience

Nowadays, the problem of cyber defense is not only issue of protection of infrastructure. The very new concept is the resilience [14].The concept of resilience is applied to a lot of contexts as well as to IT infrastructures. For instance, in the ecology the resilient item is a kind of population property that is able to survive. This concept is used in economics, construction industry [15] etc.

In the term of cybersecurity and it usage for IT infrastructure, the concept of resilience is the special ability to execute the resistance, restoration and adaption in the situation of cyberattacks performing [18-20].

So, the importunate of the development of IT infrastructure resilience is very significant.

Techniques for resilient IT infrastructures construction

Today there is a huge number of approaches devoted to cybersecurity. But task of cyber resilience is not solved yet. Nevertheless, researches make attempts to bring new techniques to construct the IT infrastructures more resilient against the cyber threads.

In [21] the technique for resilient cyber-physical systems construction is proposed. It is devoted to the task of detection of the communication delays in the infrastructure networks in the situation of denial-of-service (DoS) attacks.

The approach previously was based on the usage of multiagent systems (MASs) in order to identify the DoS attacks. The next-gen approach is based on the distributed resilient technique, that involves as the mathematical tools technique a general heterogeneous linear multiagent systems. It enables the possibility to deal with the nonuniform communication delays.

The core of the approach is the usage of the kinds of observers, that are sampled-based. To make the approach more efficient the authors proposed to make the observers be adaptive and distributed. This idea was achieved by using a buffer mechanism, that made it possible to eliminate the heterogeneous behavior generated by communication delays. In the terms of the adopting of the adaptive distributed resilient observers, the techniques made it possible to develop the resilient mechanism able to resist the denial-of-service attacks. In addition, authors included a time-varying sampling period sequence in order to prevent the attack implementation and its detection by the sampling period of the possibly infected infrastructure.

For the purpose to verify the overall technique efficiency, using the provided resilient observers, a special system controller for detection was developed. Its implementation and the series experiments conducting has proved the effectiveness of the prosed technique for resilient cyber-physical systems.

An approach for resilient control of cyber-physical systems was proposed in [22]. It is a technique for the resilient Cyber-Physical Systems construction.

In the research the authors tried to simulate different situations that may cause with IT infrastructures during its functioning. The core of the idea is to develop the system that must support the correct operations set for the crucial functional elements notwithstanding providing the resistant misbehavior.

The technique uses a moving target defense paradigm. The main idea is to deal with the linear switching of state-space matrices. The approach involves both the physical and network layers concerning a control system presented in network.

The efficiency of the proposed technique was substantiated by the set of experiments. The results have showed that appliance of the technique for the systems had made it possible to maintain systems' stability. We also evaluate, via simulation, a step-by-step procedure that takes a transfer function, representing the dynamics of the physical process.

In addition, author proved that the involvement of the approach made it possible to develop the resilient IT infrastructure, where there is a topology of decentralized controllers.

In [23] an approach for the constructing of the resilient systems against the cyberattack is proposed. Authors presented that today there is a strong need to protect the infrastructures under the attacks. To do this the cybersecurity issue must be organized around the main terms of confidentiality, integrity and availability. In addition, the main problem and drawbacks of cybersecurity of the IT- infrastructures is its strong increasing. In this situation, the cybersecurity for the IT- infrastructures has become unable to take into account the huge aspects as the time dynamics of time, space bound behavior, rapid changes etc.

In the approach the authors proposed the analysis of construction aspects concerning the resilient systems under the cyberattack.

The main idea is that there is a need to make the cyberattack resilient missions, that will give the opportunities to achieve the completion of resilient mission goals. Also, it will give the possibility to provide the IT assets and the needed services, which will enable the support of the resilient actions concerning the attacked system.

The research presents also a set of architectural issues in order to construct proper cyberattack resilience missions. It involves the mission-centricity, survivability (using the adaptation procedure). In addition, it includes the mission C2, cybersecurity management mission to achieve the efficient mission execution.

To perform the evaluation of the effectiveness of the proposed approach, the authors have developed the resilient IT infrastructure, that includes two multi-agent systems. These systems are adaptive and have possibility to

interact. Furthermore, researchers presented a set of models and algorithms, defined for the proposed resilient system with the experimental results.

In [24] an approach for the resilient cyber-physical systems construction is presented. It includes the set of steps for the development of an updated state estimation process, that is aimed to increase the resilience of IT-infrastructures under the cyberattacks.

Authors of the technique proposed to involve the existing data, that are presented in the current state estimators. The main idea of the approach is to establishes the state estimation vulnerability concerning the cyberattacks that are able to execute the data injection target at evading detecting by the mean of the outlier routines, that are used in the situation of the estimation processes.

In addition, the researchers presented the set of architectural solutions based on usage of the emerging smart grid technologies.

In [25] an agent-based cyber control strategy design for resilient control systems was produced. It presents an approach directed into defense of the critical infrastructure.

Mentioned approach includes several sets:

1) development of the cyber security research built into the industrial control system environment. The system involves the set of mechanisms to present opened and closed loop, feedback, designs.

2) integration of the cyber-physical design. It has to ensure the possibility to utilize the system data for correct response on the physical system.

3) integration of the techniques that are able to address a new approach to distributed IT infrastructures. , which considers both the industrial process control dynamics for SES, as well as the influences of the benign and malicious human.

In [26] a technique for resilient cyber-physical systems construction was presented.

In the research the authors proposed several contributions to ensure cybersecurity and infrastructure resilience. To do this the framework WAMPAC was developed. It is the security frame work, tht makes it possible to provide the end-to-end attack-resilient IT infrastructure in the power grid.

The approach involves the cybersecurity issues:

1) framework maintains the infrastructure life cycle (such as risk assessment, cyberattacks prevention measures, cyberattacks detection procedures, cyberattacks mitigation measures);

2) framework involves a defense-in-depth principles that combines the combines the cyberattacks resilience at IT infrastructure and the software levels;

3) framework is able to detect the cyber–physical security anomalies.

The authors also have presented a set of cyberattack-resilient algorithms, such as anomaly detection and mitigation approaches. The paper includes the set of experiments that prove the efficiency of the frame work. To do this it presents some cases how to prevent, detect and mitigate the attacks.

In [27] research devoted to the theoretical and some practical issues concerning the resilience of IT infrastructures are presented.

The paper shows the vast importance of the resilience today, as it can increase the adversary's effort level for the needed for the malicious objectives achievement. In terms of resilient infrastructure, it must be highly resistant to malware activities and can be able to prevent the cyberattacks execution.

The author proposed to construct systems, that were able to evaluate resilience in the presence of cyberattacks. The research involves the game-based simulation framework, that demonstrates the process of attack as well as the defending procedure.

An approach also shows a set of simulations of sufficient fidelity. To do this the manuscript includes the description of complex heterogeneous simulations. The framework is modeling integration tool names SURE for infrastructure against the cyberattacks.

The inbuild modeling simulator uses the models, constructed with the use of a model-based integration tool the heterogeneous and distributed simulations. It is able to perform the construction of the rapid design, synthesis, and evaluation of experiments.

The main feature of the SURE framework is the possibility to make the transportation systems. In this case the framework is able to provide the needed domain-specific languages, specified models, tools for the translation of constructed models. The proposed framework has in-built simulation driver tool. It's main aim is to perform the establishment of a coherent experimentation environment.

Conclusions

The paper presents a state-of-art of the methods for cyberattacks detection in the computer networks. The main accent was made on the concept of the resilience for the IT infrastructure. The concept of cyber resilience in the terms of cybersecurity was presented. The survey includes the set of approaches devoted to the problem of construction resilient infrastructures. All investigated approaches are aimed to construct and maintain infrastructure's resilience for cyberattacks resistance. Mentioned techniques and frameworks keep the main principles to assure resilience. To do this there exists some requirements to construct such infrastructure:

1) IT infrastructure has to include the set ready to use measures of preparation concerning the possible cyber threats;

2) it must include the set of special measures for the protection, as well as for the cyberattacks detection;

3) important issue and required is the possibility to respond the attack and to be able to absorb the negative attacks' impact;

4) IT infrastructure must be as adaptive as it is possible, because today the dynamic of the attacks mutation is very high;

5) IT infrastructure must be recoverable after the attacks were performed.

In addition, the state-of-art found out that known approaches have domain-specific usage and it is important to develop new approaches and frameworks for the cyberattacks detection in the computer networks as a means of resilient IT-infrastructure construction.

References

1. McAfee Mobile Threat Report Q2, 2021. URL: https://www.mcafee.com/content/dam/cons umer/en-us/docs/2020-Mobile-Threat-Report.pdf. (accessed 10.10.2021).
2. S. Liu, B. Xu, S. Li and Y. Liu, Resilient control strategy of cyber-physical system under DoS attacks, *36th Chinese Control Conference (CCC)*, 2017, pp. 7760-7765, doi: 10.23919/ChiCC.2017.8028584.
3. X. Koutsoukos et al., SURE: A Modeling and Simulation Integration Platform for Evaluation of Secure and Resilient Cyber–Physical Systems, *Proceedings of the IEEE*, vol. 106, no. 1, pp. 93-112, Jan. 2018, doi: 10.1109/JPROC.2017.2731741.
4. A. M. Melin, E. M. Ferragut, J. A. Laska, D. L. Fugate and R. Kisner, A mathematical framework for the analysis of cyber-resilient control systems, *6th International Symposium on Resilient Control Systems (ISRCS)*, 2013, pp. 13-18, doi: 10.1109/ISRCS.2013.6623743.
5. L. Zhang, X. Chen, F. Kong and A. A. Cardenas, Real-Time Attack-Recovery for Cyber-Physical Systems Using Linear Approximations, *IEEE Real-Time Systems Symposium (RTSS)*, 2020, pp. 205-217, doi: 10.1109/RTSS49844.2020.00028.
6. A. Lukina, A. Tiwari, S. A. Smolka, L. Esterle, J. Yang and R. Grosu, Resilient Control and Safety for Cyber-Physical Systems, *IEEE Workshop on Monitoring and Testing of Cyber-Physical Systems (MT-IT INFRASTRUCTURES)*, 2018, pp. 16-17, doi: 10.1109/MT-IT INFRASTRUCTURES.2018.00015.
7. C. Deng and C. Wen, MAS-Based Distributed Resilient Control for a Class of Cyber-Physical Systems With Communication Delays Under DoS Attacks, *IEEE Transactions on Cybernetics*, vol. 51, no. 5, pp. 2347-2358, May 2021, doi: 10.1109/TCYB.2020.2972686.
8. S. Sulochana and V. Manjula, Resilient system for secure sharing of information against false data injection attack, *International Conference on Information Communication and Embedded Systems* (ICICES), 2016, pp. 1-5, doi: 10.1109/ICICES.2016.7518943.
9. Lysenko, S., Bobrovnikova, K., Savenko, O., Shchuka, R. Technique for Cyberattacks Detection Based on DNS Traffic Analysis, *CEUR-WS*, Vol 2732. ISSN: 1613–0073. 2020. pp. 171-182
10. Bobrovnikova, Kira, Sergii Lysenko, Piotr Gaj, Dmytro Denysiuk "Technique for IoT Cyberattacks Detection Based on DNS Traffic Analysis.CEUR-WS.Vol.2623. ISSN: 1613–00732020, pp. 208-218.
11. Sergii Lysenko, Kira Bobrovnikova, Peter Popov, Viacheslav Kharchenko, Dmytro Medzatyi. Spyware Detection Technique Based on Reinforcement Learning. *CEUR-WS*.Vol. 2623 ISSN: 1613–0073 (2020), pp. 307-316
12. Lysenko, S., Bobrovnikova, K., Matiukh, S., Hurman, I., Savenko, O.Detection of the botnets' low-rate DDoS attacks based on self-similarity. *International Journal of Electrical and Computer Engineering*, ISSN 2088-8708, 2020, 10(4), pp. 3651-3659.
13. Savenko, O., Sachenko, A., Lysenko, S., Markowsky, G., Vasylkiv, N. Botnet detection approach based on the distributed systems. *International Journal of Computing*, ISSN 1727-6209, 2020, 19(2), pp. 190-198.
14. Sergii Lysenko, Kira Bobrovnikova, Peter Popov, Viacheslav Kharchenko, Dmytro Medzatyi. Spyware Detection Technique Based on Reinforcement Learning. CEUR-WS.Vol. 2623 ISSN: 1613–0073 (2020), pp. 307-316
15. S. Al-Rabiaah, The "Stuxnet" Virus of 2010 As an Example of A "APT" and Its "Recent" Variances, *21st Saudi Computer Society National Computer Conference (NCC)*, 2018, pp. 1-5, doi: 10.1109/NCG.2018.8593143.
16. Lysenko S., Bobrovnikova Y., Kharchenko V. Methods of detecting botnets in computer systems. *Modern information systems*. 2019. T.3. №4. C.87-95.
17. Sergii Lysenko, Kira Bobrovnikova, Piotr Gaj, Tomas Sochor, and Iryna Forkun Resilient Computer Systems Development for Cyberattacks Resistance. *CEUR-WS*. 2021, 2853, pp. 353–361.
18. Cimellaro G. P., Dueñas-Osorio L., Reinhorn A.M. Introduction to special issue on resilience-based analysis and design of structures and infrastructure systems. Structural Engineering, 2016, No. 142(8), pp.1-5.
19. Linkov, I., Eisenberg, D. A., Plourde, K., Seager, T. P., Allen, J., Kott, A. Resilience metrics for cyber systems. Environment Systems and Decisions, 2013, No. 33(4), pp. 471-476.
20. Bodeau, D.J., Graubart, R. D. Cyber resiliency design principles: selective use throughout the lifecycle and in conjunction with related disciplines. MITRE Corp., Tech. Rep., 2017, p.98.
21. C. Deng, Distributed Resilient Control for Cyber-Physical Systems under Denial-of-Service Attacks, *23rd International Conference on Mechatronics Technology (ICMT)*, 2019, pp. 1-5, doi: 10.1109/ICMECT.2019.8932108.
22. M. Segovia-Ferreira, J. Rubio-Hernan, R. Cavalli and J. Garcia-Alfaro, Switched-Based Resilient Control of Cyber-Physical Systems, *IEEE Access*, vol. 8, pp. 212194-212208, 2020, doi: 10.1109/ACCESS.2020.3039879.
23. G. Jakobson, Mission-centricity in cyber security: Architecting cyberattack resilient missions, *5th International Conference on Cyber Conflict* (CYCON 2013), 2013, pp. 1-18.
24. S. Hopkins, E. Kalaimannan and C. S. John, Foundations for Research in Cyber-Physical System Cyber Resilience using State Estimation, *SoutheastCon*, 2020, pp. 1-2, doi: 10.1109/SoutheastCon44009.2020.9249745.
25. C. Rieger, Q. Zhu and T. Başar, Agent-based cyber control strategy design for resilient control systems: Concepts, architecture and methodologies, *2012 5th International Symposium on Resilient Control Systems*, 2012, pp. 40-47, doi: 10.1109/ISRCS.2012.6309291.
26. A. Ashok, M. Govindarasu and J. Wang, Cyber-Physical Attack-Resilient Wide-Area Monitoring, Protection, and Control for the Power Grid, *Proceedings of the IEEE*, vol. 105, no. 7, pp. 1389-1407, July 2017, doi: 10.1109/JPROC.2017.2686394.
27. X. Koutsoukos et al., SURE: A Modeling and Simulation Integration Platform for Evaluation of Secure and Resilient Cyber–Physical Systems, *Proceedings of the IEEE*, vol. 106, no. 1, pp. 93-112, Jan. 2018, doi: 10.1109/JPROC.2017.2731741.

Sergii Lysenko **Сергій Лисенко**	Doctor of Science, Professor of Computer Engineering & System Programming Department, Khmelnytskyi National University, Khmelnytskyi, Ukraine, e-mail: sirogyk@ukr.net, orcid.org/0000-0001-7243-8747, Scopus Author ID: 54420643500, ResearcherID: I-1728-2018 https://scholar.google.com.ua/citations?hl=uk&user=TuAfytwAAAAJ&view_op=list_works	доктор технічних наук, професор кафедри комп'ютерної інженерії та системного програмування, Хмельницький національний університет, Хмельницький, Україна
Dmytro Sokalskyi **Дмитро Сокальський**	Master student, Computer Engineering, Khmelnytskyi National University, Khmelnytskyi, Ukraine, e-mail: sokalskij7@gmail.com	студент, комп'ютерна інженерія, Хмельницький національний університет, Хельницький, Україна.
Iana Mykhasko **Яна Михасько**	Master student, Computer Engineering, Khmelnytskyi National University, Khmelnytskyi, Ukraine, e-mail: yashamyy@gmail.com	студентка, комп'ютерна інженерія, Хмельницький національний університет, Хельницький, Україна.

Додаток В

(обов'язковий)

Копія тез доповіді на Всеукраїнськійнауково-практичнійконференції АктуальніПроблемиКомп'ютерних Наук (АПКН-2021)

УДК 004.7.056.5

Сокальський Д. О., Лисенко С. М.

Хмельницький національний університет

ПРОГРАМНО-ТЕХНІЧНИЙ ЗАСІБ ВИЯВЛЕННЯ ВТОРГНЕНЬ В IT-ІНФРАСТРУКТУРИ

Запропоновано програмно-технічний засіб виявлення вторгнень в IT-інфраструктури на основі аналізу мережного трафіку корпоративних мереж IT-інфраструктур на різних рівнях мережі. В роботі запропоновано моделі функціонування програмно-технічного засобу виявлення вторгнень, а також відомих атак на IT-інфрастуктури. Запропонований метод здійснює аналіз поведінки трафіку засобами системи виявлення в нормальних і відмовних умовах, коли атаки відображаються на збоях функціонування комп'ютерної мережі.

A software and hardware tool for detecting intrusions into IT infrastructure based on the analysis of network traffic of corporate networks of IT infrastructures at different levels of the network is proposed. The paper proposes models of the operation of software and hardware intrusion detection, as well as known attacks on IT infrastructure. The proposed method analyzes the behavior of traffic by means of the detection system in normal and negative conditions, when the attacks are reflected in the failure of the computer network.

Зі стрімким зростанням загроз безпеці IT-інфраструктурам, програмно-технічні засоби виявлення атак, які здатні відстежувати діяльність комп'ютерних мереж або окремих хостів на предмет зловмисної поведінки, є незамінним компонентом безпеки сучасних корпоративних мережі [1-3].

Найбільш поширеними програмно-технічними засобами виявлення атак є системи виявлення вторгнень (IDS) [2,4].

Серед найбільш поширених підходів до побудови системи виявлення вторгнень є такі методи проектування IDS: на основі підписів, які можуть виявляти лише відомі атаки, та системи на основі виявлення аномалій, які можуть виявляти як відомі, так і невідомі атаки, але генерують велику кількість помилкових спрацювань [5-7].

Існують такі класи атак, як атака на основі ARP, атака на основі ICMP, низькошвидкісна DoS-атака тощо, які уникають виявлення як за підписом, так і аномаліями [8-9]. Тому актуальною є задача розроблення програмно-технічних засобів виявлення вторгнень в IT-інфраструктури.

В дослідженні пропонується розроблення програмно-технічного засобу системи виявлення вторгнень в IT-інфраструктури на основі аналізу мережного трафіку корпоративних мереж IT-інфраструктур на різних рівнях мережі.

В роботі запропоновано моделі функціонування програмно-технічного засобу виявлення вторгнень, а також відомих атак на IT-інфраструктури.

Запропонований метод здійснює аналіз поведінки трафіку засобами системи виявлення в нормальних і відмовних умовах, коли атаки відображаються на збоях функціонування комп'ютерної мережі.

В основі роботи методу лежить оцінка стану, яка отримується в результаті діагностування мережі. Підсистема діагностування здійснює відстежування послідовності подій, які генерує система, щоб вирішити, чи відповідають стани, через які система проходить, нормальній або несправній моделі функціонування мережі на різних рівнях.

Ядром роботи системи виявлення вторгнень в IT-інфраструктури є діагностування мережі, а для виявлення атак на основі ARP використовується активний механізм зондування на основі запитів та відповідей ARP.

Для вирішення задачі було застосовано активний фреймворк DES, який використовується для моделювання атак на основі ARP з використанням контрольованої події (зонд ARP), що створює різницю в послідовності подій для нормальних умов або умов атаки на IT-інфраструктуру.

Крім того, для виявлення низькошвидкісних атак типу TCP DoS атака необхідним є впровадження аналізу справжньої послідовності станів з такими, які мають відхилення в значеннях.

Тому для виявлення цієї атаки метод застосовує стохастичний підхід, який здатний з певною ймовірністю ідентифікувати випадок атаки.

Враховуючи міграцію з адресації IPv4 на IPv6 в Інтернеті, було запропоновано механізм виявлення атак мережі IPv6 на основі NDP.

Для перевірки ефективності запропонованого методу було здійснено ряд експериментальних досліджень, результати яких показують достатньо високу ефективність виявлення втогнень в IT-інфраструктури на рівні понад 93% з рівнем хибних спрацювань виявлення 4%.

Перелік посилань

1. M. Kumar, A. K. Singh, Distributed Intrusion Detection System using Blockchain and Cloud Computing Infrastructure, 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), 2020, pp. 248-252, doi: 10.1109/ICOEI48184.2020.9142954.
2. Лисенко С. М., Харченко В.С., Бобровнікова К.Ю., Щука Р. Резильєнтність комп'ютерних систем в умовах кіберзагроз: Онтологія та таксономії. Радіоелектронні і комп'ютерні системи. 2020. №1. С. 17-28.

3. Лисенко С. М. Моделі кібератак мережного та хостового типу. Вимірювальна та обчислювальна техніка в технологічних процесах. 2019. №2. С. 58-63.
4. A. Borkar, A. Donode and A. Kumari, A survey on Intrusion Detection System (IDS) and Internal Intrusion Detection and protection system (IIDPS), 2017 International Conference on Inventive Computing and Informatics (ICICI), 2017, pp. 949-953, doi: 10.1109/ICICI.2017.8365277.
5. P. Widulinski, K. Wawryn, A Human Immunity Inspired Intrusion Detection System to Search for Infections in an Operating System, 2020 27th International Conference on Mixed Design of Integrated Circuits and System (MIXDES), 2020, pp. 187-191, doi: 10.23919/MIXDES49814.2020.9155771.
6. J. Yu, P. Tian, H. Feng, Y. Xiao, Research and Design of Subway BAS Intrusion Detection Expert System, 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2018, pp. 152-156, doi: 10.1109/IAEAC.2018.8577262.
7. S. OUIAZZANE, M. ADDOU, F. BARRAMOU, "A Multi-Agent Model for Network Intrusion Detection," 2019 1st International Conference on Smart Systems and Data Science (ICSSD), 2019, pp. 1-5, doi: 10.1109/ICSSD47982.2019.9003119.
8. A. Warzyński and G. Kołaczek, "Intrusion detection systems vulnerability on adversarial examples," 2018 Innovations in Intelligent Systems and Applications (INISTA), 2018, pp. 1-4, doi: 10.1109/INISTA.2018.8466271.
9. Z. S. Malek, B. Trivedi, A. Shah, User behavior Pattern -Signature based Intrusion Detection, 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), 2020, pp. 549-552, doi: 10.1109/WorldS450073.2020.9210368.

Додаток Г

(обов’язковий)

Презентація доповіді

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
Кафедра комп’ютерної інженерії та системного програмування

Сокальський Дмитро Олександрович

Програмно-технічний засіб виявлення вторгнень в ІТ-інфраструктури

Науковий керівник – д.т.н. доц. Лисенко С.М.

Хмельницький - 2022

Мета і задачі дослідження

Метою роботи є підвищення достовірності та ефективності виявлення вторгнень в ІТ-інфраструктури.

Об'єкт дослідження – процес вторгнень в ІТ-інфраструктури.

Предмет дослідження – моделі, удосконалений метод та програмно-технічні засоби виявлення вторгнень в ІТ-інфрастуктури.

Мета і задачі дослідження

Поставлена мета досягається розв’язанням наступних задач:

- дослідити особливості функціонування програмно-технічних засобів систем виявлення вторгнень в ІТ-інфраструктури;
- проаналізувати сучасні програмно-технічних засоби виявлення вторгнень в ІТ-інфраструктури;
- розробити модель процесу виявлення вторгнень в ІТ-інфраструктури
- розробити модель функціонування програмно-технічних засобів виявлення вторгнень в ІТ-інфраструктури, виконати моделювання обробки підроблених пакетів при атаці на ІТ-інфраструктуру, побудувати модель атак на ІТ-інфраструктуру з вимірюваністю, побудувати модель атак на ІТ-інфраструктуру з керованістю;
- розробити удосконалений метод виявлення вторгнень в ІТ-інфраструктури, що ґрунтується на застосуванні DES-моделей;
- реалізувати програмно-технічні засоби виявлення вторгнень в ІТ-інфраструктури.

Наукова новизна отриманих результатів

1. Удосконалено метод виявлення вторгнень в IT-інфраструктури, який на відміну від відомих ґрунтується на застосуванні DES-моделей, які характеризуються дискретним простором станів і певною динамікою, керованою подіями: в нормальних умовах, а також за кожного з умов атаки.
2. Набули подальшого розвитку програмно-технічних засобів виявлення вторгнень в IT-інфраструктури, які забезпечують виявлення вторгнень з високою достовірністю та ефективністю.

Практичне значення отриманих результатів

Практичне значимість реалізованого програмно-технічного засобу виявлення вторгнень в IT-інфраструктури продемонструвало підвищення достовірності та ефективності точності виявлення вторгнень в IT-інфраструктури до 97%.

Відомі методи, на яких ґрунтуються системи виявлення вторгнень

- IDS на основі підпису
- IDS на основі аномалій

Актуальність дослідження

- Дослідження надзвичайно актуальне у теперішній час. Адже кожного дня сотні атак на IT-інфраструктури проводяться зі сторони країни-агресора.

- Завдяки програмно-технічній базі може бути дана можливість забезпечення безперебійної та безпечної роботи усіх складових IT-сектору.

- Збільшена точність виявлення загроз зменшує час та затрати на їх виявлення, впродовж всього терміну користування IT-структурами.

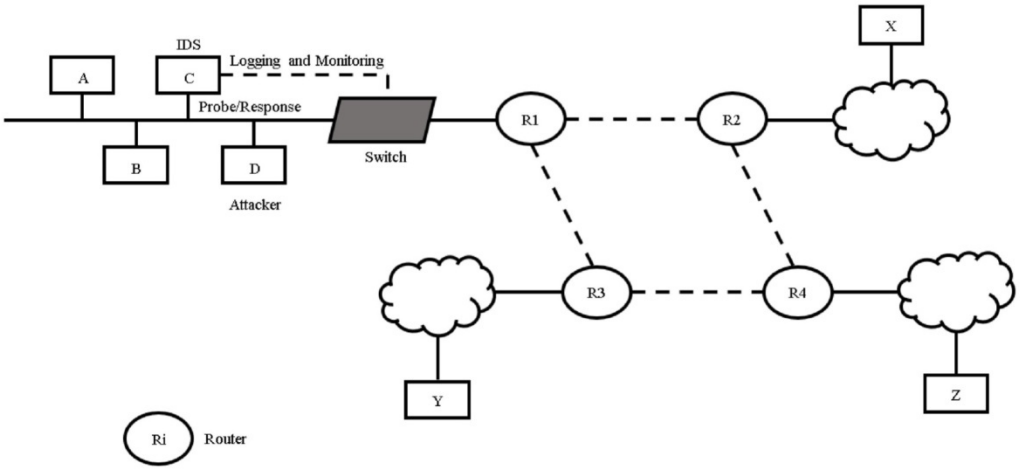
УДОСКОНАЛЕНИЙ МЕТОД ВИЯВЛЕННЯ ВТОРГНЕНЬ В IT-ІНФРАСТРУКТУРИ

Етапи методу

1. Здійснення моніторингу мережі
2. Застосування механізму активного зондування
3. Перевірка правильності пакетів
4. Обробка підроблених пакетів
5. Моделювання DES в нормальних умовах та в умовах здійснення вторгнення
6. Застосування детектора

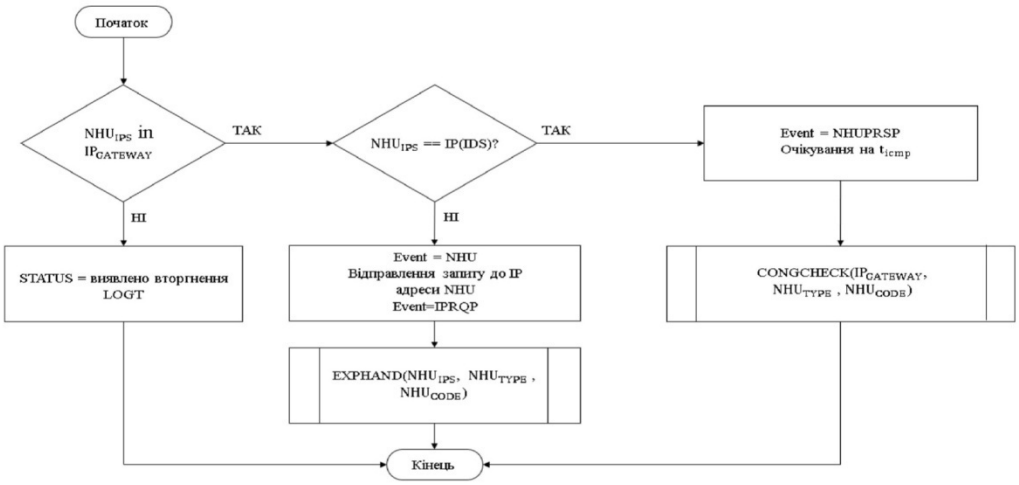
Здійснення моніторингу мережі

Для ілюстрації схеми використовується архітектура мережі, показана на рисунку



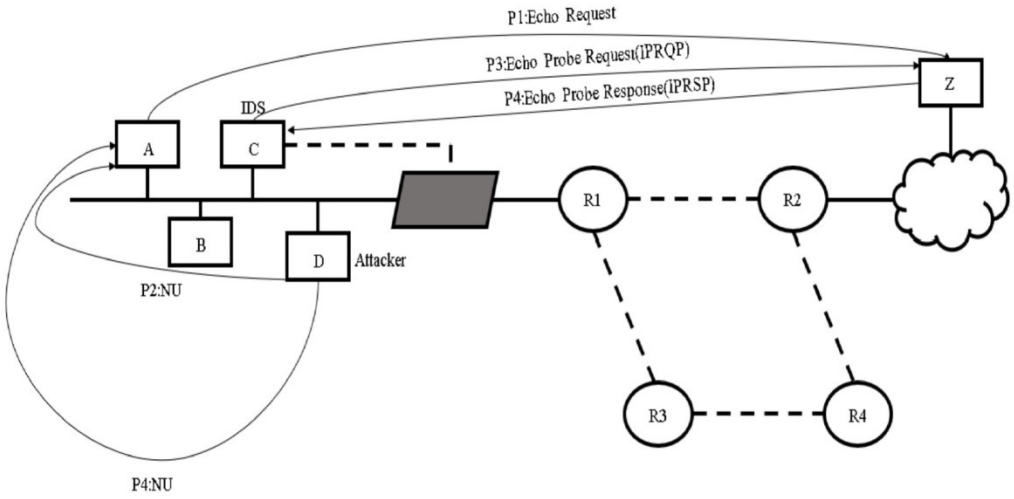
Детектор DES для атаки NHU

Перевірка правильності пакетів



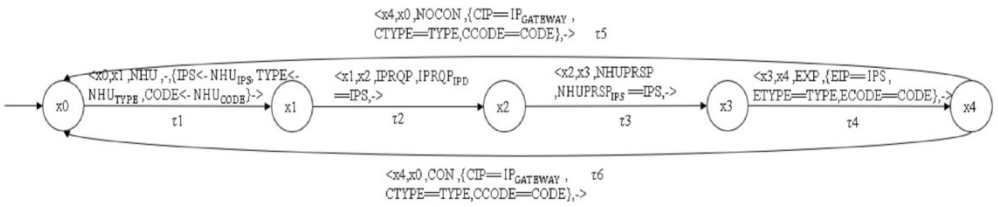
Блок-схема для обробника недоступних мереж/хосту

Обробка підроблених пакетів

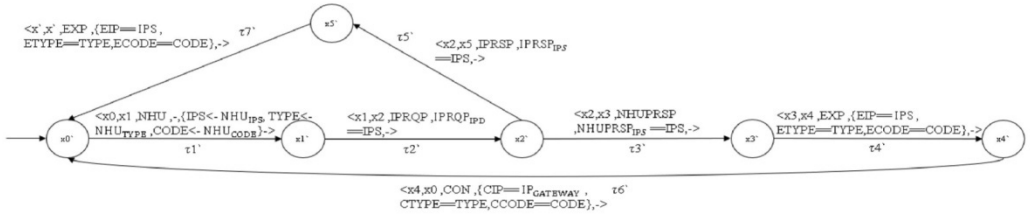


Приклад для мережі/хост недоступний

Моделювання DES в нормальних умовах та в умовах здійснення вторгнення

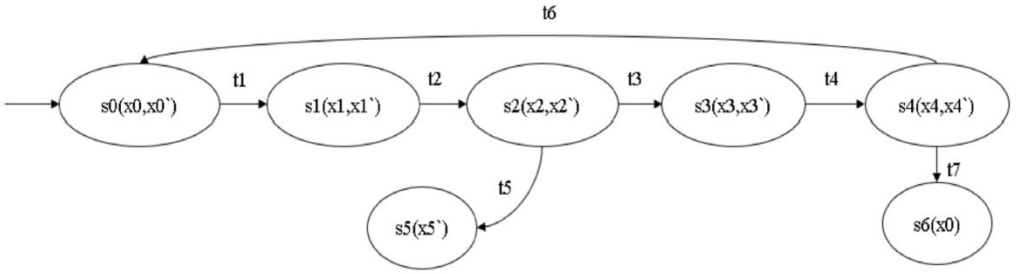


Модель DES в нормальному стані



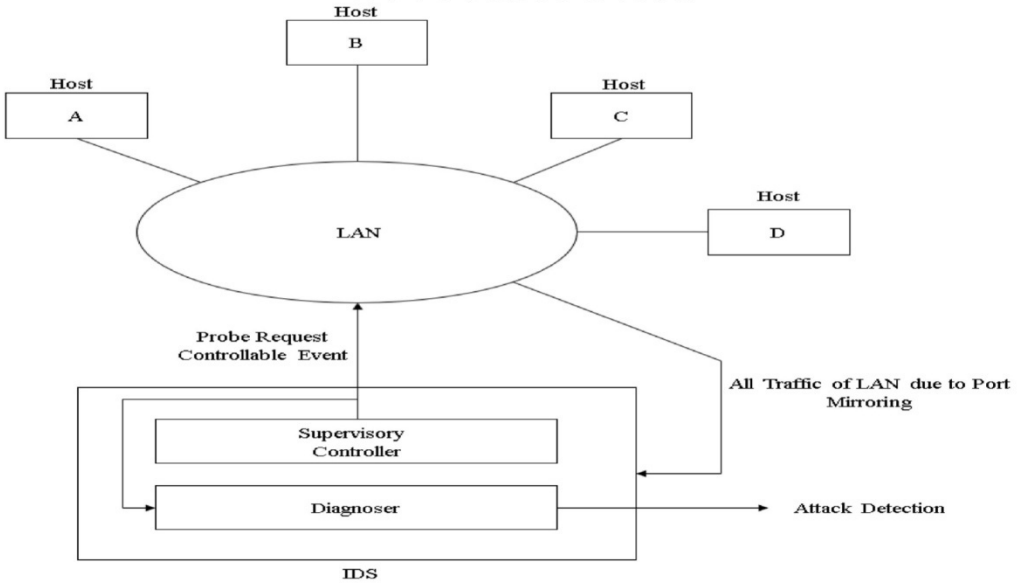
Модель DES в умовах атаки

Застосування детектора

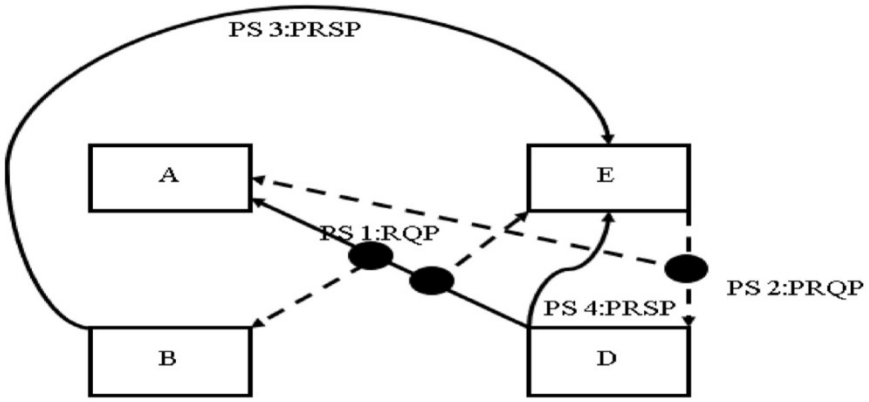


I-детектор для моделей DES на попередньому слайді

ПРОГРАМНО-ТЕХНІЧНИЙ ЗАСІБ ВИЯВЛЕННЯ ВТОРГНЕНЬ В IT-ІНФРАСТРУКТУ



ПРОГРАМНО-ТЕХНІЧНИЙ ЗАСІБ ВИЯВЛЕННЯ ВТОРГНЕНЬ В IT-ІНФРАСТРУКТУ



Приклад підробленого запиту

IT-інфраструктура для проведення експериментальних досліджень

Випробувальний стенд був створений для експериментального аналізу запропонованої IDS.
IT-інфраструктура складається з 6 комп'ютерних систем під керуванням операційних систем – Windows 10 (як робочі станції), openSUSELeap 15.3 (як маршрутизатор), openSUSELeap 15.3 (як робочі станції), КС з Kali 2022.1 є комп'ютерною системою-зловмисником, а також окрема комп'ютерна система під керуванням openSUSELeap 15.3 налаштована як IDS.
Ці комп'ютерні системи підключаються за допомогою комутатора CISCO Catalyst серії 3560 G з увімкненим дзеркальним відображенням портів для програмно-технічного засобу виявлення вторгнень в IT-інфраструктури.

Дослідження швидкості, точності засобами запропонованого програмно-технічного засобу виявлення вторгнень в ІТ-інфраструктури

% залучених пакетів	Кількість атак	Рівень виявлення (%)
100	1000	97.22
90	1000	85.36
80	1000	83.12
70	1000	77.55
60	1000	73.77
50	1000	69.33
40	1000	61.34
30	1000	59.69
20	1000	57.09
10	1000	51.23

Статистика підробки ARP

Публікації за матеріалами дипломної роботи

- За темою дипломної роботи опубліковано статтю на тему «Дослідження методів виявлення кібератак в комп'ютерних мережах як засобів до побудови резильєнтних до вторгнень ІТ-інфраструктур» в фаховому журналі «Комп'ютерні системи та інформаційні технології» №3 за 2021 рік [1], а також опубліковано тези у Всеукраїнській науково-практичній конференції Актуальні Проблеми Комп'ютерних Наук (АПКН-2021) [2]. Було взято участь у Всеукраїнській науково-практичній конференції Актуальні Проблеми Комп'ютерних Наук.

Зв’язок роботи з науковими програмами, планами, темами

- Дослідження, представлені у кваліфікаційній роботі, проводились в рамках держбюджетної НДР Хмельницького національного університету 1Б-2021 «Самоорганізована розподілена система виявлення зловмисного програмного забезпечення в комп’ютерних мережах» (ДР № 0121U109936) 2021-2022 рр.

Висновки

- Магістерська робота присвячена розв’язанню науково-практичної задачі підвищення достовірності та ефективності точності виявлення вторгнень в IT-інфраструктури.
- В першому розділі було досліджено системи виявлення вторгнень. Також було проаналізовано таксономію IDS на основі виявлення атак, зокрема IDS на основі підпису та IDS на основі аномалій . було проаналізовано обмеження існуючих IDS.
- В розділі також досліджено основні аспекти функціонування вторгнень засобами атаки на основі ARP, атаки через повідомлення ICMP, засобами повільних DoS-атак, а також атак на основі NDP.
- Виявлено недоліки відомих методів програмно-технічних засобів систем виявлення вторгнень в IT-інфраструктури.

Висновки

- Зроблено висновок та постановку задачі щодо необхідності удосконалення методу виявлення вторгнень в IT-інфраструктури.
- В другому розділі запропоновано моделювання процесу виявлення вторгнень в IT-інфраструктури. З цією метою було застосовано апарат дискретно-подійне моделювання (DES). Вказаний математичний апарат дозволив здійснити наступні дослідження, а саме виконати моделювання обробки підроблених пакетів при атаці на IT-інфраструктуру, побудувати модель атак на IT-інфраструктуруз вимірюваністю., побудувати модель атак на IT-інфраструктуру з керованістю, побудувати моделювання відмов, побудувати DES-модель атаки типу спуфінгу ARP.

Висновки

- В третьому розділі було запропоновано удосконалений метод виявлення вторгнень в IT-інфраструктури, що ґрунтується на застосуванні DES-моделей, які характеризуються дискретним простором станів і певною динамікою, керованою подіями в нормальних умовах, а також за кожного з умов атаки. В основі методу лежить механізм оцінки стану (детектор), який спостерігає за послідовністю подій, створених системою, щоб вирішити, чи відповідають стани, через які система проходить, нормальній чи несправній моделі DES, і далі застосовується механізм активного зондування.Також в розділі було представлено, наприклад, виявлення вторгнення атакою NHU. Метод містить наступні кроки: моніторинг мережі, застосування механізму активного зондування, перевірка правильності пакетів, обробка підроблених пакетів, моделювання DES в нормальних умовах та в умовах здійснення вторгнення, застосування детектора.

Висновки

- В четвертому розділі представлено реалізацію удосконаленого методу було реалізовано програмно-технічний засіб виявлення вторгнень в IT-інфраструктури, який використовує DES на основі І-діагностики. Запропонований IDS дозволяє виявити атаки шляхом діагностики лише на тих шляхах, де за несправністю слідує індикаторна подія.
- Практичне застосування реалізованого програмно-технічного засобу виявлення вторгнень в IT-інфраструктури продемонструвало підвищення достовірності та ефективності точності виявлення вторгнень в IT-інфраструктури до 97%.

UNICHECK

Ім'я користувача:
Кафедра КІ

Дата перевірки:
29.04.2022 08:50:29 EEST

Дата звіту:
29.04.2022 08:50:53 EEST

ID перевірки:
1010993731

Тип перевірки:
Doc vs Internet + Library

ID користувача:
100005591

Назва документа: Програмно-технічний засіб виявлення вторгнень в IT-інфраструктури

Кількість сторінок: 96 Кількість слів: 21713 Кількість символів: 144601 Розмір файлу: 917.47 KB ID файлу: 1010898854

3.93%
Схожість

Найбільша схожість: 2.46% з Інтернет-джерелом (https://iiti.ac.in/people/~neminath/Papers/ISA.pdf)

3.05% Джерела з Інтернету	44	Сторінка 98
1.37% Джерела з Бібліотеки	105	Сторінка 98

0.19% Цитат

Цитати	2	Сторінка 99

Не знайдено жодних посилань

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	52

Fri Apr 29 07:59:41 EEST 2022, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 1.0%

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 10%**

	Документ		Суммарное совпадение по Базе Данных	
ID: 103209 Название: Програмно-технічний засіб виявлення вторгнень в IT-інфраструктури Добавлено в БД: 2022-04-29 Авторы: Сокальський Д.О. Руководители: Лисенко С.М. Консультанты: Оппоненты:	Символы	Лексемы	Символы	Лексемы
	131658	1090	1252 (1%)	10 (1%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА ДИПЛОМНУ РОБОТУ

Дипломник: Сокальський Дмитро Олександрович

Тема: Програмно-технічний засіб виявлення вторгнень в IT-інфраструктури

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг дипломної роботи:

Кількість листів креслень 0; кількість сторінок записки 93

1. Короткий зміст роботи та прийнятих рішень У роботі запропоновано Програмно-технічний засіб виявлення вторгнень в IT-інфраструктури

2. Висновок про відповідність роботи дипломному завданню Дипломна робота відповідає виданому завданню

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: В результаті реалізації удосконаленого методу було реалізовано програмно-технічний засіб виявлення вторгнень в IT-інфраструктури.

Програмно-технічний засіб виявлення вторгнень в IT-інфраструктури використовує DES на основі I-діагностики. У схемі використовується активний механізм для виявлення атак без використання дорогого криптографічного механізму. Також схема не вимагає зміни в стеку протоколів і не порушує архітектуру шарів.

Запропонований IDS дозволяє виявити атаку шляхом діагностики лише на тих шляхах, де за несправністю слідує індикаторна подія. Це дозволяє виявляти атаки ICMP, незважаючи на такі проблеми, як перевантаження мережі.

Атаки ICMP можна виявити, розділивши звичайну послідовність подій атаки, шляхом діагностики певних шляхів за допомогою I-діагностики DES.

Практичне застосування реалізованого програмно-технічного засобу виявлення вторгнень в IT-інфраструктури продемонструвало підвищення достовірності та ефективності точності виявлення вторгнень в IT-інфраструктури до 97%.

4. Позитивні сторони роботи: В результаті виконаного наукового дослідження було розроблено програмно-технічних засобів виявлення вторгнень в IT-інфраструктури, які забезпечує виявлення вторгнень з високою достовірністю та ефективністю.

5. Негативні сторони роботи: Не в повній мірі здійснено аналіз методів атак на основі NDP.

6. Оцінка графічного оформлення та пояснювальної записки роботи Матеріали кваліфікаційної роботи є структурованими у чіткій та логічній формі та відображають послідовність виконання поставлених задач

7. Відгук про роботу в цілому: В загальному робота виконана на відмінному рівні.

8. Інші зауваження: –

9. Оцінка дипломної роботи:

Розглянувши позитивні та негативні сторони представленої дипломної роботи вважаю, що робота заслуговує оцінки «відмінно»

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) Джулій В.М. , к.т.н., доцент, кафедри кафедри кібербезпеки та комп'ютерних систем і мереж Хмельницького національного університету

" 27 " квіня 2022р.

Завідувачу кафедри КІІС
д-р.техн.наук, проф. Говорущенко Т. О.

Сокальський Дмитро Олександрович

ПІБ здобувача вищої освіти

ФІТ, 2 курсу, групи КІ2м-20-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіатоповіщений (а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

27.04.2022

дата

підпис

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА СИСТЕМНОГО ПРОГРАМУВАННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Програмно-технічний засіб виявлення вторгнень в ІТ-інфраструктури

Автор: Сокальський Дмитро Олександрович

Спеціальність: 123 – Комп'ютерна інженерія та програмування

Освітня програма: освітньо-наукова

Науковий керівник: Лисенко С.М., д.т.н, професор

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є незначними, законними і не є плагіатом, оскільки:

1) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з 105 джерелами з бібліотек та 44 джерелами з мережі Інтернет;
2) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 3.93% і адресується до 44 першоджерел, а найбільша схожість становить 2.46% що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ С.М. Лисенко

Гарант ОП _____ О. С. Савенко

Завідувач кафедри КІСП _____ Т. О. Говорущенко