БАКАЛАВРСЬКА РОБОТА

БР. ІП - 84.00.00.000 ПЗ

Група ІП-22-3

Суп Ігор

2026

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Суп Ігор Віталійович

(прізвище, ім'я, по батькові)

УДК 004.942
(індекс)

БАКАЛАВРСЬКА РОБОТА

Розробка програмного забезпечення для симуляції кібератак та тренування захисних команд

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня _____Суп І.В._____
(підпис, ініціали та прізвище здобувача)

Науковий керівник _____Ваврик Тетяна Олександрівна, асистент
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри

_доц._____Бандура В.В.__
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2026

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
доцент. В.В. Бандура
“ ____ ” _____2026 р.

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Суп Ігор Віталійович

(прізвище, ім’я, по-батькові)

1. Тема проекту (роботи) "Розробка програмного забезпечення для симуляції кібератак та тренування захисних команд"

керівник проекту (роботи) Ваврик Тетяна Олександрівна, асистент

затверджені наказом вищого навчального закладу від “ 21_ ” _квітня 2026 р. № ___179/7____

2. Строк подання студентом проекту (роботи) 10 червня 2026 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження переддипломної практики

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

1. Аналіз вимог до програмного забезпечення
2. Аналіз вимог і постановка задачі
3. Проектування системи
4. Реалізація проекту
5. Тестування та впровадження

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1 Орієнтований графік пріоритету стадій (рис. 1.1, ст. 19)

2 Рівні злиття JDL (рис. 1.2, ст. 21)

4. Функціональність симулятора кібератаки (рис. 2.1, ст. 29)

4 Панель сценаріїв атаки на мережевому фреймі (рис. 3.1, ст. 45)

5 Приклад розвитку атаки (рис. 3.3, ст. 48)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

2. Дата видачі завдання **2026 р.**

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Визначення та обгрунтування теми роботи	17.01.2026	виконано
2	Огляд існуючих концепцій, рішень та сервісів в даній області	21.02.2026	виконано
3	Побудова моделі або алгоритму власного рішення	01.03.2026	виконано
4	Документування реалізації власного оригінального рішення вибраними засобами	21.04.2026	виконано
5	Оформлення пояснювальної записки бакалаврської роботи	02.05.2026	виконано

Студент _____Суп І.В.
(підпис)

Керівник роботи _____Ваврик Т.О.
(підпис)

АНОТАЦІЯ

Бакалаврська робота містить 90 сторінок, 35 рисунків, 7 таблиці, список використаних джерел із 40 найменувань.

Метою роботи є дослідження принципів розробки програмного забезпечення для симуляції кібератак та тренування захисних команд.

Об'єкт дослідження: програмні системи симуляції кібератак і платформи для тренування команд кіберзахисту.

Предмет дослідження: методи, технології та інструменти моделювання кібератак, побудови сценаріїв атак, обробки подій безпеки, моніторингу інцидентів та оцінки ефективності захисних команд.

Результати дослідження: проведено аналіз сучасних підходів до симуляції кібератак, розглянуто методи моделювання атак і механізми реагування на інциденти, а також виконано оцінку ефективності рішеня.

У першому розділі розглянуто теоретичні основи кібербезпеки, основи забезпечення захисту інформаційних систем, класифікацію та моделювання кібератак, а також сучасні підходи до побудови симуляційних середовищ.

У другому розділі досліджено методи та інструменти створення програмного забезпечення для симуляції кібератак, архітектуру програмного пакета та методології моделювання атак і кіберінцидентів.

У третьому розділі розглянуто процес розробки та інтеграції програмного забезпечення, реалізацію сценаріїв атак, механізми обробки подій безпеки, а також проведено оцінку ефективності роботи системи та аналіз результатів експериментів.

Висновок: використання програмних платформ для симуляції кібератак дозволяє підвищити рівень підготовки захисних команд, покращити навички реагування на кіберінциденти та підвищити загальний рівень безпеки.

КЛЮЧОВІ СЛОВА: КІБЕРБЕЗПЕКА, КІБЕРАТАКИ, КІБЕРПОЛІГОН, СИМУЛЯЦІЯ АТАК, ІНФОРМАЦІЙНА БЕЗПЕКА, SIEM, IDS/IPS, SOC, МОДЕЛЮВАННЯ ІНЦИДЕНТІВ, ТРЕНУВАННЯ ЗАХИСНИХ КОМАНД.

ANNOTATION

The thesis consists of 90 pages, 35 figures, 7 tables, and a reference list containing 40 sources.

The aim of the work is to study the principles of developing software for cyberattack simulation and defensive team training.

Research object: software systems for cyberattack simulation and cybersecurity team training platforms.

Research subject: methods, technologies, and tools for cyberattack modeling, attack scenario creation, security event processing, incident monitoring, and evaluation of defensive team performance.

Research results: modern approaches to cyberattack simulation were analyzed, the principles of cyber range construction were studied, methods of attack modeling and incident response mechanisms were reviewed, and the effectiveness of software solutions for cybersecurity team training was evaluated.

The first section describes the theoretical foundations of cybersecurity, principles of information system protection, classification and modeling of cyberattacks, and modern approaches to building simulation environments.

The second section analyzes methods and tools for developing cyberattack simulation software, including the structure of simulation models, software package architecture, and methodologies for modeling attacks and cyber incidents.

The third section covers the development and integration of the software solution, implementation of attack scenarios, security event processing mechanisms, and evaluation of system effectiveness and experimental results.

Conclusion: the use of software platforms for cyberattack simulation improves the preparedness of defensive teams, helps identify vulnerabilities in information systems, and increases the overall level of organizational cybersecurity.

KEY WORDS: CYBERSECURITY, CYBERATTACKS, CYBER RANGE, ATTACK SIMULATION, INFORMATION SECURITY, SIEM, IDS/IPS, SOC, INCIDENT MODELING, DEFENSIVE TEAM TRAINING.

ЗМІСТ

					БР.ІП –84.00.00.000 ПЗ			
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розроб.		Суп І.В.			Розробка програмного забезпечення для симуляції кібератак та тренування захисних команд **Пояснювальна записка**	*Літ.*	*Арк.*	*Акрушів*
Перевір.		Ваврик Т.О.					7	
Реценз.		Процюк Г.Я.				**ІФНТУНГ ІП-22-3**		
Н. Контр.		Піх М.М.						
Затверд.		Бандура В. В.						

ВСТУП

У сучасному цифровому середовищі кібербезпека стала одним із ключових факторів забезпечення стабільності та безперервності роботи інформаційних систем. Активне впровадження хмарних технологій, Інтернету речей, автоматизованих систем управління та розподілених мереж призводить до постійного збільшення кількості кіберзагроз і складності атак. Організації різних сфер діяльності стикаються з необхідністю оперативного реагування на інциденти інформаційної безпеки, захисту критичної інфраструктури та підготовки фахівців, здатних ефективно протидіяти сучасним кібератакам. У таких умовах особливої актуальності набуває створення програмних рішень для симуляції кібератак і тренування захисних команд у контрольованому та безпечному середовищі.

Актуальність теми зумовлена зростанням кількості складних кібератак, необхідністю підготовки висококваліфікованих спеціалістів із кібербезпеки та потребою у створенні сучасних кіберполігонів для моделювання реальних сценаріїв атак. Існуючі програмні рішення для навчання та тестування систем безпеки часто мають обмежену функціональність, недостатню масштабованість або складність інтеграції з сучасними інструментами моніторингу та аналізу подій. Крім того, важливим викликом є забезпечення реалістичності симуляцій, автоматизація створення сценаріїв атак і можливість аналізу дій команд реагування. Розробка спеціалізованого програмного забезпечення для симуляції кібератак дозволить підвищити ефективність навчання фахівців, покращити навички реагування на інциденти та виявляти потенційні вразливості інформаційних систем.

Метою роботи є розробка програмного забезпечення для симуляції кібератак та тренування захисних команд, яке забезпечує моделювання сценаріїв атак, моніторинг подій безпеки, аналіз дій користувачів та оцінку ефективності реагування на кіберінциденти.

Завданнями дослідження є аналіз предметної області та існуючих рішень для симуляції кібератак, визначення функціональних і нефункціональних вимог до програмного забезпечення, дослідження сучасних методів

моделювання атак і засобів кіберзахисту, проектування архітектури системи, реалізація модулів генерації сценаріїв атак і обробки подій, тестування програмного забезпечення та оцінка його ефективності, а також формулювання висновків щодо перспектив використання системи у сфері кібербезпеки.

Об'єктом дослідження є процеси розробки програмного забезпечення для симуляції кібератак і тренування захисних команд, що включають аналіз вимог, моделювання сценаріїв кіберінцидентів, реалізацію механізмів моніторингу та оцінку ефективності роботи системи.

Предметом дослідження є методи, технології та програмні інструменти, що використовуються для побудови систем симуляції кібератак, моделювання кіберзагроз, аналізу подій безпеки, автоматизації процесів навчання та тестування захисних команд.

Методи дослідження включають аналіз наукових джерел для вивчення сучасних підходів у сфері кібербезпеки, порівняльний аналіз платформ симуляції атак, методи моделювання кіберінцидентів, експериментальний метод для реалізації та тестування програмного забезпечення, а також статистичний аналіз результатів роботи системи.

Розроблене програмне забезпечення передбачатиме можливість створення та запуску сценаріїв кібератак, моніторингу подій безпеки, аналізу мережевої активності, оцінки дій захисних команд і формування звітів щодо результатів тренувань. Особлива увага приділятиметься масштабованості системи, безпеці даних, автоматизації процесів моделювання та зручності використання програмного інтерфейсу. Очікується, що впровадження системи дозволить підвищити рівень підготовки спеціалістів із кібербезпеки, покращити швидкість реагування на інциденти та сприятиме підвищенню захищеності інформаційних систем.

Бакалаврська робота містить 90 сторінок, 35 рисунків, 7 таблиць, 3 розділи, список використаних джерел із 40 найменувань.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ СИСТЕМ СИМУЛЯЦІЇ КІБЕРАТАК

1.1 Основи кібербезпеки та принципи побудови кіберполігонів

Комп'ютерні мережі значно розвинулися за останні роки. Від мереж коледжів до бізнес-мереж, майже всі аспекти їхньої функціональності покращилися як щодо швидкості, так і щодо ємності. Через ці покращення збільшення розміру та складності таких мереж стало звичайним явищем. Це зростання розміру та складності ускладнює завдання точного моніторингу мережевої активності та підтримки всіх бажаних стандартів безпеки для мережевих адміністраторів.

Надійність, яку люди та організації очікують від мереж, також різко зросла за останні роки. Спостерігається зростаюча залежність від мереж для зберігання необхідної інформації та забезпечення багатьох операцій, від яких покладаються окремі особи та організації. Для деяких організацій інформація та операції, що надаються через мережу, є важливими для функціонування організації. Від банківських фірм до комунікаційних компаній та охорони здоров'я існує сильна залежність від комп'ютерних мереж для забезпечення інформації та функціональності, необхідних цим галузям.

Через великий розмір, складність та залежність від комп'ютерних мереж, проблеми безпеки зростають [1]. Кібератаки на такі комп'ютерні мережі стають дедалі поширенішими, а загрози цих атак зростають, оскільки організації продовжують зберігати та отримувати доступ до критично важливої інформації через комп'ютерні мережі. Атаки через Інтернет є особливо серйозною проблемою, і вважається, що на них припадає близько 70% усіх зловмисних атак на організації. Крім того, виникнення внутрішніх та віддалених атак стає менш поширеним як через збільшення інтернет-активності організацій, так і через легкість атак через інтернет-з'єднання. Також, зі збільшенням трафіку в самій мережі, конкретні дії атаки все більше приховуються в цьому трафіку. Це може посилити труднощі, з якими стикаються адміністратори та сенсорні

пристрої в ефективному виявленні цих атак.

Кібератаки стають легшими для менш навчених хакерів завдяки великій кількості корисних програм та наявності інформації, пов'язаної з організацією та мережею, яку можна отримати навіть за допомогою простого пошуку в Google. Оскільки організації залишають конфіденційну інформацію доступною для користувачів поза своєю мережею, пошукові системи, такі як Google, можуть знаходити цю інформацію (за допомогою ботів) та відображати її будь-якому зацікавленому хакеру, який може запропонувати відповідні критерії пошуку [2]. Крім того, інструменти та програми, що використовуються для вторгнень, стають потужнішими, вимагаючи менше знань від зловмисників, які їх використовують. Наприклад, операції підбору паролів, які раніше виконувалися вручну, тепер автоматизуються за допомогою різноманітних інструментів. Такі інструменти стають більш доступними для широкої хакерської спільноти, тому зловмисникам більше не потрібно бути експертами в цій галузі.

Однак ці вразливі комп'ютерні мережі не залишилися беззахисними. Системи виявлення вторгнень (IDS) відіграли певну роль, допомагаючи мережевим адміністраторам знаходити та відстежувати мережеві атаки. Виявлення вторгнень пропонує засоби, за допомогою яких мережевий трафік може контролюватися пристроєм або датчиком, окрім моніторингу адміністратором. Це робиться за допомогою датчика, який визначає, чи може певний пакет мережевого трафіку виконати або дозволити потенційно шкідливу дію. Якщо так, генерується сповіщення. IDS може надавати попередження, щоб вказати на виявлення шкідливої або підозрілої активності, і ці попередження можуть допомогти системі та адміністраторам захиститися від атаки. Крім того, IDS може перевірити, чи зміни в конфігураціях безпеки мережі дійсно забезпечують додаткову безпеку. Сьогодні доступні кілька різних типів систем виявлення вторгнень, починаючи від безкоштовного програмного забезпечення з відкритим кодом і закінчуючи дорогими, спеціалізованими та високоінтегрованими власницькими системами [4].

Хоча сповіщення датчиків справді вказують на підозрілі події, все ще існує потреба переглядати ці сповіщення (або деякі з них) та намагатися

визначити, що насправді відбувається в мережі. Саме в цьому процесі ідея об'єднання інформації нещодавно була впроваджена в кіберсферу. Об'єднання інформації – це техніка, за якої дані з багатьох різних датчиків поєднуються з відповідними базами даних інформації, щоб зробити висновки, які неможливо було б зробити, дивлячись лише на окремі датчики. Об'єднання даних кількох датчиків у кіберсфері – це досить нова концепція, яка передбачає об'єднання даних з кількох датчиків IDS для розуміння мережевої активності та подій. У кіберсфері визначення намірів та загрози зловмисника шляхом розгляду дій окремо є складним завданням [3]. Використання об'єднання інформації як інструменту дозволяє аналізувати багато дій разом, щоб отримати загальне розуміння поточної мережевої ситуації. Високий рівень хибних тривог систем виявлення вторгнень (та пов'язані з ними проблеми), а також недостатня здатність більшості систем виявлення вторгнень отримати повне розуміння поточної ситуації в мережі, підкреслюють необхідність покращення способів обробки цих сповіщень датчиків. Використовуючи методи об'єднання інформації, існує потенціал для розробки нового покоління надійних систем виявлення вторгнень, які надаватимуть ситуаційну інформацію на додаток до інформації про окремі дії.

Навіть з розробкою таких систем виявлення вторгнень з використанням методів об'єднання інформації, все ще існує складність тестування та перевірки того, що системи забезпечують бажану точність та функціональність. Отримання значних обсягів даних датчиків IDS є складним завданням, а налаштування фізичних мереж для проведення таких тестів вимагає значного часу та коштів [5]. Крім того, мінливість мереж, а також цінність інформації, що зберігається в таких мережах, можуть зробити процес врахування різних мережевих структур практично неможливим для обробки за допомогою фізичної конфігурації. Розробка систем виявлення вторгнень з використанням об'єднання інформації потребує значної кількості надійних даних для перевірки та підтвердження продуктивності систем .

1.2 Методи забезпечення інформаційної безпеки та підготовки захисних команд

Забезпечення безпеки в приватній мережі є одним з головних завдань, з якими стикаються організації щодо управління інформацією. Компанії часто зберігають конфіденційну та особисту інформацію в приватній мережі, і ці компанії роблять це з метою, щоб доступ до інформації мали лише надійні особи. Однак часто хакери (з особистими мотивами або пов'язаними з конкуруючими організаціями) вважають таку інформацію дуже цінною та вдається проникнути в приватну мережу, щоб отримати або пошкодити потрібну інформацію. Більшість приватних мереж мають певну форму підключення до Інтернету, що робить веб-атаки загрозою. Хакери відстежують такі мережі та визначають вразливості та експлойти, які допоможуть їм отримати доступ до таких мереж. Хоча заходи безпеки в приватних мережах удосконалюються, пошук усіх вразливостей та підтримка різних заходів безпеки в актуальному стані є складним завданням для адміністраторів мереж компанії, і ті, хто планує атакувати приватну мережу, розраховують саме на це. Крім того, нові експлойти завжди будуть знайдені до того, як будуть застосовані заходи безпеки для боротьби з ними. Через складність підтримки безпеки у світі приватних мереж, важливість, яку приватні мережі та інформація, що зберігається в них, мають для компаній та організацій у всьому світі, а також частоту, з якою хакери атакують такі мережі для отримання бажаної інформації, предметом цього дослідження буде безпека в приватних мережах [6].

Розробка інструментів мережевої безпеки, що використовують методи об'єднання інформації, є значним кроком уперед у надійному захисті приватної мережі.

Однак, як зазначалося раніше, у цій сфері ще багато чого потрібно зробити через складність розробки надійних систем об'єднання інформації та перевірки того, чи точно вони розпізнають атаки. При об'єднанні інформації з кількох датчиків IDS критичним фактором є розуміння того, чи достатньо добре

розроблюваний механізм об'єднання виявляє атаки, що відбуваються. Дані про атаки та датчики, що надаються на етапах розробки механізмів об'єднання інформації, можуть значно допомогти розробникам підтвердити їхню функціональність. Щоб перевірити, чи достатньо добре виявляються атаки, можна налаштувати мережу, де виконуються відомі атаки, а датчики збирають пов'язані сповіщення про відповідні дії. Однак цей варіант створює значні витрати для розробників, і така схема також не враховує велику мінливість намірів та стилів атак, які можуть статися. З цих причин пропонується використовувати імітаційне моделювання для точного відображення як потоку атаки через мережу, так і сповіщень IDS, пов'язаних з цією атакою. Моделювання атак та генерація сповіщень IDS дозволяє швидко генерувати значні обсяги даних, пов'язаних з атаками та датчиками, з невеликими витратами або без них. Крім того, це моделювання дозволить легко моделювати мережі різного розміру та структури та використовувати їх для розробки даних про атаки та датчики. Надаючи змодельовані дані про кібератаки, розробники систем об'єднання інформації матимуть можливість протестувати свої механізми об'єднання в широкому діапазоні сценаріїв.

Розроблено модель та методологію моделювання для отримання даних про атаки та дані датчиків, що використовуються системами термоядерного злиття. Ця модель моделювання враховує різні розміри та конфігурації мережі, але існують значні обмеження в точному зображенні приватних мереж та атак на ці мережі. Метою цієї роботи є надання моделі моделювання та методології, яка враховує обмеження та проблеми існуючої моделі та покращує точність і продуктивність моделі у генерації достовірних даних про атаки та дані датчиків [7].

Існує багато важливих критеріїв для розробки коректної моделі симуляції атаки з високою точністю та високою продуктивністю. До них, загалом, належать можливість легкого запуску та зміни сценаріїв атаки, забезпечення значного контролю над змодельованою мережею, створення достовірних та детальних даних, пов'язаних з атаками та системами виявлення вторгнень (ISS), а також можливість використання кількох застосувань для моделювання систем виявлення вторгнень. Що стосується простоти зміни та запуску сценаріїв, модель

симуляції повинна надавати користувачеві інтуїтивно зрозумілі елементи керування, пропонувати варіанти щодо того, як вихідні дані повинні генеруватися та оброблятися, а також графічно відображати аспекти мережі та атаки. Враховуючи всі ці міркування, буде менше ймовірності неправильного використання результуючої моделі або генерування даних, які користувач не мав наміру генерувати. Забезпечуючи значний контроль над змодельованою мережею, симуляція повинна дозволяти широкий спектр деталей на розсуд користувача. Це включає як топологію мережі, так і атрибути окремих пристроїв [8]. Наприклад, симуляція повинна надавати можливість вказувати вразливості мережевого пристрою, які відображатимуть доступні дії атаки проти пристрою. Крім того, симуляція повинна дозволяти легко додавати датчики IDS та інструменти IPS до мережевих пристроїв. Наявність цих опцій дозволить моделі краще відображати незначні відмінності, що існують між реальними мережевими пристроями. Щодо створення достовірних та детальних мережевих даних, створені атаки та згенеровані сповіщення повинні бути точно відображені. Дані датчиків IDS повинні бути репрезентативними для виходу, який надають фактичні датчики [9]. Атаки, створені симулятором, повинні відповідати логічній послідовності кроків атаки на основі існуючих змодельованих шаблонів атак. Щоб модель симуляції була цінною для багатьох застосувань при моделюванні систем виявлення вторгнень, модель повинна мати налаштовані параметри для обробки різноманітної визначеної користувачем інформації та методи, за допомогою яких можна додавати та використовувати нові (або різні) датчики IDS у моделі.

1.3 Моделювання, класифікація та сценарії кібератак у тренувальних системах

Ефективним засобом розуміння кібератак є класифікація атак та моделювання їхнього розвитку. Це передбачає ретельне спостереження за діями, пов'язаними з відомою атакою, щоб зрозуміти поведінку зловмисника.

На щастя, у моделюванні атак досягнуто значного прогресу.

Наприклад, розроблено адаптивну систему моделювання кібератак для допомоги в тестуванні захисту програмного забезпечення. Раніше команда експертів у предметній області (SME) мала імітувати дії, зазвичай пов'язані з хакерськими атаками, щоб оцінити та перевірити свою методологію захисту програмного забезпечення. Однак цей метод значно збільшує вартість і час, необхідні для такого проекту, і він насправді не підходить для великої кількості сценаріїв, які потребують ретельного тестування. Дослідження, проведене Догерті та Гонслейвзом, показало, що розробка точних моделей кібератак може суттєво зменшити вартість і час, необхідні для цього проекту. За допомогою цього моделювання було визначено три основні категорії атак: атаки на веб-додатки (вважаються найбільш вразливими через значну кількість інструментів, доступних хакеру), атаки клієнт-серверних додатків та атаки на окремі системи (які найскладніше реалізувати). Крім того, моделі реалізували баєсівський підхід мережі переконань для імітації міркувань хакерів [10]. Хоча моделювання, проведене в рамках цього дослідження, охоплювало лише те, що було необхідно для тестування захисту програмного забезпечення, методологія добре відображає те, що стає необхідним для будь-якого типу інструментів безпеки.

Інші дослідники досліджували моделювання атак, аналізуючи окремі сповіщення IDS, пов'язані з кроками атаки. Наприклад, розробили проект під назвою «Моделювання корельованих атак» (CAM), де сценарії атак моделюються шляхом спостереження за фактичними сповіщеннями IDS. Ці сповіщення IDS зазвичай можна пов'язати з певним кроком атаки (зазвичай певним експлойтом). Хоча це дозволяє використовувати фактичні атаки для розробки шаблонів атак, багато сповіщень IDS можуть бути хибнопозитивними та, таким чином, впливати на процес моделювання [11]. Крім того, деякі кроки атаки можуть бути взагалі пропущені або можуть бути настільки розподілені в часі, що складання цілої атаки (не кажучи вже про додавання до шаблону атаки) стає складним завданням.

Також було проведено дослідження вищого рівня для визначення типової послідовності різних експлойтів, які може виконати хакер. Розробили шаблон на основі графів, який використовує методи теорії графів для позначення

типів атакуючих дій або експлойтів, необхідних перед тим, як можна буде виконати інші певні типи. Ця розробка почалася з групування відомих експлойтів у категорії або «етапи» на основі того, який тип діяльності був необхідний до того, як ці експлойти відбулися, і який тип діяльності міг бути виконаний після цього. Потім була розроблена матриця суміжності для співвіднесення різних етапів, що вказує на те, які етапи можуть відбутися після того, як відбувся експлойт з заданого етапу для всіх етапів у шаблоні. Спрощений орієнтований граф цього шаблону експлойту атаки показано на рисунку 1.1.

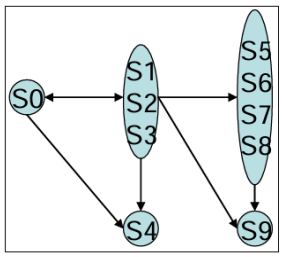

Рисунок 1.1 - Орієнтований графік пріоритету стадій

Вузли, що містять S0 - S9, представляють десять етапів експлойтів атаки, а ребра (стрілки) представляють пріоритет етапів. Наприклад, експлойт типу вторгнення з етапу 1 (де зловмисник отримує доступ користувача до пристрою) відбудеться лише після виконання експлойту типу розвідки з етапу 0 (де зловмисник отримує корисну інформацію про пристрій). Класифікуючи можливі експлойти відомих вразливостей за одним із етапів, процес моделювання атаки спрощується тим, що дійсно потрібно моделювати лише етапи атаки, на відміну від величезної кількості експлойтів, які потрібно було б враховувати в іншому випадку. Завдяки цим перевагам цей шаблон на основі графів був використаний у моделі моделювання, щоб дозволити автоматичне
моделювання атак на основі набору параметрів.

Методи об'єднання інформації, що застосовуються в кіберсфері

Методи об'єднання інформації, які нещодавно застосовувалися до питань кібербезпеки, мають потенціал для виявлення корисних тенденцій та отримання

значної кількості інформації про хакерські атаки на основі даних, що надаються датчиками IDS [12]. Загалом, багатосенсорне об'єднання даних стосується методів, що використовуються для об'єднання даних з кількох датчиків разом із відповідною інформацією з баз даних та шаблонів для визначення певної інформації про систему, що цікавить. Використання кількох датчиків дозволяє отримувати точнішу інформацію, яка враховує ширший спектр системи, на додаток до простої статистичної переваги наявності більшої кількості точок даних для прийняття рішення. Цей метод об'єднання інформації раніше використовувався в механізмах спостереження, наведення та контролю транспортних засобів, моніторингу техніки, медичної діагностики та інших військових та невійськових цілях.

Загальний процес об'єднання даних має п'ять конкретних рівнів функціональності, визначених Спільним комітетом директорів лабораторій (JDL). На рисунку 1.2 зображено ці рівні стосовно різних застосувань об'єднання даних з кількох датчиків.

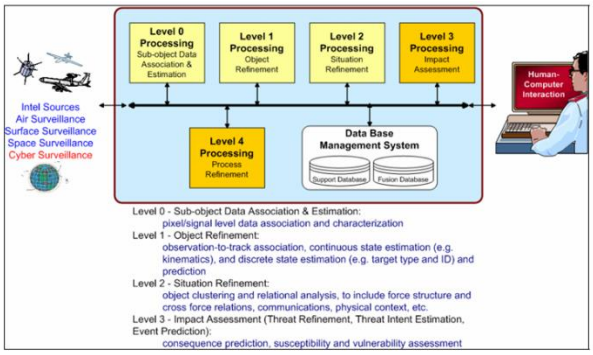

Рисунок 1.2 - Рівні злиття JDL

Рівень 0 стосується попередньої обробки даних, де дані, що найбільше стосуються поточної ситуації, фільтруються з великого набору необроблених даних. Рівень 1 представляє уточнення об'єкта, де дані трансформуються для забезпечення узгодженості, а також ідентифікуються та класифікуються кілька атрибутів об'єктів. Рівень 2 представляє уточнення ситуації, де взаємозв'язки між об'єктами та подіями розробляються шляхом включення інформації про навколишнє середовище та попередніх (апріорних) знань [13]. Рівень 3 представляє

уточнення загроз, де поточна ситуація проектується на майбутнє для визначення потенційних загроз та вразливостей системи, а також наявних можливостей. Рівень 4 представляє уточнення процесу, де контролюється довгострокова ефективність об'єднання даних, виявляються можливості для покращення та вносяться корективи до системи для досягнення бажаних ефектів.

«Механізм об’єднання інформації для прийняття рішень у режимі реального часу» (так званий INFERD) з необхідними можливостями використовується з даними сповіщень IDS у кібердомені. По суті, сповіщення, що генеруються датчиками IDS, обробляються як точки даних, які INFERD може приймати як вхідні дані для об'єднання інформації для виявлення атак або шкідливої активності, що відбувається в мережі [14]. Таким чином, INFERD може надати певну інформацію про поточний стан мережі (у режимі реального часу), з якої отримано дані сповіщень. Що стосується рівнів об'єднання JDL, INFERD враховує комбінацію об'єднання рівня 1 та рівня 2. На додаток до вхідних даних у механізм, INFERD також вимагає шаблонів атак разом з деякою інформацією щодо топології мережі. Хоча метою цього механізму є співвіднесення фактичних сповіщень для визначення атак, процес все ще перебуває в розробці та потребує ретельного тестування.

Оскільки фактичні дані сповіщень IDS отримати складно, як обговорювалося раніше, і оскільки не існує бездоганного методу оцінки того, чи двигун ідеально ідентифікує ситуацію, використання моделювання для отримання репрезентативних даних сповіщень IDS є дуже корисним у цьому спектрі процесу об'єднання. Крім того, допомагаючи розробці INFERD, використання моделювання також може забезпечити переваги для вищих рівнів цього процесу об'єднання. Наприклад, TANDI, двигун об'єднання, що використовується на рівні 3 процесу об'єднання, надає прогнози щодо наступних кроків зловмисника (по суті, загрози, що створюється хакером). За допомогою моделі моделювання ми точно знатимемо, які наступні кроки зловмисника, і ці знання можна буде порівняти з результатами, які надає TANDI, щоб оцінити ефективність TANDI. Моделювання може навіть вийти за рамки цього, щоб дозволити двигуну або методології,

зосередженій на вдосконаленні процесу (об'єднання рівня 4), оновлювати модель моделювання в режимі реального часу та оцінювати, наскільки добре функціонує вся система об'єднання в мережі [15].

Об'єктно-орієнтоване симуляційне моделювання

Використання об'єктно-орієнтованого підходу до симуляційного моделювання має багато переваг порівняно з процедурним підходом. Можливість повторного використання та розширення об'єктів є центральною особливістю та перевагою розробки об'єктно-орієнтованого фреймворку. Крім того, оскільки об'єкти є модульними, інформація, пов'язана з кожним екземпляром об'єкта, зберігається лише в одному місці, з посиланнями на розташування об'єкта за потреби [16]. Крім того, за допомогою об'єктно-орієнтованого моделювання, що зосереджено на представленні об'єктів, обчислення та логіку моделювання можна розділити між об'єктами (класами). Таке делегування функцій моделювання забезпечує більш організовану структуру, ніж це було б можливо за допомогою процедурного моделювання.

Першим кроком у розробці об'єктно-орієнтованої моделі моделювання є розробка відповідної концепції архітектури програмного забезпечення. Це включає широке визначення того, як модель та аспекти моделювання й елементи керування взаємодіятимуть на високому рівні. Сарджугян та Сінгх (2003) рекомендують використовувати три окремі фрейми в архітектурі моделювання для забезпечення необхідної взаємодії. До них належать фрейм моделі, елемент керування та фрейм перегляду. У цьому випадку фрейм моделі складає функціональність програми та обробляє стани представлених об'єктів. Елемент керування пов'язує дії користувача зі змінами моделі та вибирає конкретні «вигляди» або стани системи для представлення користувачеві. Вид представляє користувацьку сторону архітектури, де візуалізується модель та надаються жести користувача. Розробляючи архітектуру дискретно-подійного моделювання, вказується на важливість забезпечення «світового погляду на поток транзакцій», де дискретна кількість трафіку або транзакцій переміщується з точки в точку (або від об'єкта до об'єкта) всередині системи, щоб змінити стан системи. Можна застосувати

різні підходи для забезпечення такого типу взаємодії, але головне - створити структуру, яка чітко визначатиме цей потік транзакцій [17].

Структура класів у бажаному фреймворку моделювання є ще одним важливим фактором. Пропонується використовувати централізований клас моделювання, який об'єднує всі аспекти системи. Клас містить посилання на клас планувальника , який обробляє різні дискретні події, що відбуваються під час моделювання. Клас менеджера подій використовується для забезпечення реалізації різних подій у моделі моделювання [18]. Крім того, задіяні ресурси обробляються через клас ресурсів, пов'язаний з класом моделювання. Такий підхід дозволяє класу моделювання спеціально обробляти транзакції, що беруть участь у моделюванні; хоча існує певна надмірність у наявності як класу планувальника, так і класу менеджера подій. Надано детальну структуру, за допомогою якої можна налаштувати ієрархію класів моделювання. Вершина цієї структури включає абстрактний клас з властивостями та методами, доступними для всіх компонентів моделювання. Дочірні класи (підкласи) цього класу включають клас випадкових чисел, який забезпечує генерацію та розподіл випадкових чисел, клас статистики для обчислення статистики моделі на основі атрибутів, доступних абстрактному класу, та елемент моделювання для моделювання та запуску моделювання. Елемент моделювання має п'ять основних дочірніх класів, які представляють основні компоненти моделювання. Клас подій використовується для організації та реалізації подій, визначених під час запуску моделювання. Клас процесів використовується для обробки певних процесів маніпулювання ресурсами. Клас сутностей використовується для представлення різних транзакцій, що проходять через систему. Клас вузлів представляє певні точки або об'єкти в системі, до яких входять та виходять сутності. Нарешті, клас вибору обробляє логіку, необхідну для маніпулювання транзакціями в системі [19]. Цей загальний підхід є достатньо широким, щоб врахувати численні типи систем, які можна моделювати за допомогою дискретно-подійного моделювання. Крім того, структура успадкування забезпечує легкий доступ до загальних властивостей моделювання.

Однак реальна функціональність самої симуляційної моделі все ще залишається за об'єктами. Структура симуляції та ієрархія класів корисні лише тоді, коли окремим задіяним об'єктам надаються відповідні методи та атрибути, необхідні для ефективної взаємодії та зміни стану моделі. Баезнер та Ломов зосереджуються на трьох основних критеріях для забезпечення загальних потреб симуляції: клас сутностей, клас подій та властивість для збереження часу симуляції. Клас сутностей використовується для моделювання будь-яких фізичних об'єктів або процесів, що моделюються, а клас подій використовується для планування та запуску дій усіх сутностей. Час симуляції належним чином оновлюється через події. Далі розширено рекомендації щодо ієрархії класів, надаючи конкретну інформацію про залучені класи. Три класи, що становлять інтерес в ієрархії, - це класи подій, сутностей та вузлів [20]. Клас сутностей надає елементи, які потрібно переміщувати по системі для виклику змін. Клас сутностей включає методи для отримання часу створення сутності, статусу, поточного розташування та унікального ідентифікатора (ID). Клас вузлів використовується для моделювання різних вузлів або розташувань у модельованій системі. Цей клас дозволяє створювати мережу вузлів і надає методи для отримання типу вузла, ідентифікації об'єктів на поточному вузлі, переліку вузлів у системі або мережі та визначення ідентифікатора вузла. Також є методи, що запускаються, коли об'єкт входить у вузол або залишає його. Клас events містить методи для встановлення та отримання часу події, обробки події та керування події та з чим вони пов'язані.

Більшість підходів до дискретно-подійного моделювання спираються на методи обробки серії визначених подій. Як правило, події додаються до впорядкованого списку, в якому кожна подія обробляється окремо на основі часу. У цій конфігурації годинник моделювання відстежує час і просувається дискретними кроками лише тоді, коли всі події для поточного часу завершено обробку. У структурі, події витягуються з календаря та запускають відповідні транзакції в моделі. Ці транзакції (сутності) запускають події вузлів і, таким чином, дозволяють досить децентралізований метод логічної обробки кожної події. Пропонується метод обробки подій, який зосереджується на надсиланні кожної поточної події

до певної процедури обробки подій (класу або методу) для обробки логіки, пов'-язаної з цією подією. Хоча це може спростити та впорядкувати реакції на події, логіки, необхідної для перевірки процедури, можна уникнути, використовуючи функції перевантаження успадкування класів.

Було розроблено численні програми для ілюстрації та використання можливостей об'єктно-орієнтованого моделювання [21]. Одна з таких програм, YANSL, була розроблена як ілюстрація запропонованої ними архітектури об'єктно-орієнтованого моделювання. YANSL включає як транзакційні, так і ресурсні сутності, що переміщуються через мережу вузлових об'єктів. Об'єкти вузлів складаються з різноманітних типів вузлів відправлення та призначення з різною функціональністю, включаючи вузол джерела (для створення сутностей), вузол черги (для зберігання сутностей), вузол активності (для зміни сутностей) та вузол приймача (для видалення сутностей). Обробка подій запускає методи у вузлах відправлення та вузли призначення будуть належним чином активовані та коригувати рух і атрибути сутностей у моделі. YANSL все ще має значні обмеження моделювання та статистики, але широке використання успадкування класів ефективно зменшує кількість логічних операторів і рішень, що необхідно приймати, і дозволяє децентралізовано керувати симуляцією.

Симулятор кібератаки

Модель моделювання, відома як Симулятор кібератаки, була створена за допомогою ARENA, комерційного програмного забезпечення для моделювання моделювання, розробленого Rockwell Software. У цьому програмному забезпеченні надається спеціальний шаблон для представлення мережевих пристроїв, які потрібно було змоделювати. Цей шаблон включає роз'єми (маршрутизатори або комутатори) та машини [22]. Крім того, можливість підключення цих пристроїв дозволяє користувачеві графічно налаштувати просту модель комп'ютерної мережі. На рисунку 1.3 показано приклад налаштування мережі в цьому інтерфейсі.

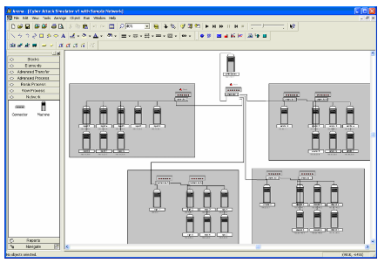

Рисунок 1.3 - Приклад комп'ютерної мережі

Машини в змодельованій мережі надаються атрибутами, такими як IP-адреса, тип доступу та використання датчика IDS, для сприяння моделюванню атаки. IP-адреса використовується як унікальний ідентифікатор, тип доступу (доступ до Інтернету) контролює, чи можуть атаки розпочатися з машини, а використання датчика IDS визначає, чи генеруватиме машина відповідні сповіщення IDS (для відповідного типу датчика) на основі трафіку до/від датчика. Також розширено ці атрибути, додавши вказівку на тип машини (чи це ПК, чи сервер) та операційну систему [23]. Ці атрибути важливі, оскільки вони звужують тип експлойтів атак, які можуть бути виконані на машині. Однак є ще кілька атрибутів, які слід враховувати для ефективного моделювання того, які експлойти можуть виникнути, особливо щодо програмного забезпечення та служб машини. Крім того, можливість представляти групу (або підмережу) машин може спростити процес моделювання мереж.

Роз'єми в змодельованій мережі також оснащені датчиком IDS, який функціонує так само, як і машини. Однак насправді існують відмінності між цими двома типами датчиків (мережеві та хостові), які слід враховувати в симуляційній моделі. Сповіщення IDS генеруються шляхом зіставлення трафіку атаки (або шуму) з сповіщенням датчика за допомогою файлу, що містить прямі кореляції між сигнатурами атаки та повідомленнями сповіщень.

Симулятор кібератаки, завдяки інтеграції з VBA, надає форми для налаштування атак на змодельовані мережі. Спочатку користувач має вказати кібератаки. Для кожного кроку вказується вихідна (атакуюча) машина та цільова (атакована) машина, а також експлойт або конкретна категорія експлойтів, що

буде використано.

Крім того, модель обмежена лише десятьма атаками та двадцятьма п'ятьма кроками на атаку. Цей процес накладає суворі обмеження на обсяг даних сповіщень IDS, які можна генерувати, та вносить значну упередженість користувача в систему. З цих причин розроблено автоматичну методологію атаки, яка генерує набір кроків атаки на основі кількох параметрів атаки, наданих користувачем. Ці параметри в першу чергу включають цільову машину, тип цілі, ефективність атаки, прихованість атаки та середній час кроку. Ефективність стосується прямості атаки та успішності кроків, тоді як прихованість стосується того, наскільки прихована атака від виявлення шляхом уникнення певних категорій експлойтів [24]. Хоча методологія забезпечила гарну основу для розробки неупереджених атак за короткий проміжок часу, все ще існують значні обмеження щодо можливих типів атак. Міркування щодо наданих параметрів та логіки в методології можуть призвести до значних покращень у процесі генерації атак.

1.4 Висновок по розділу

У першому розділі було розглянуто теоретичні основи розробки систем симуляції кібератак та тренувальних середовищ для підготовки захисних команд. Проаналізовано основи кібербезпеки та принципи побудови кіберполігонів як спеціалізованих платформ для відпрацювання сценаріїв реагування на інциденти. Досліджено методи забезпечення безпеки та підготовки фахівців у сфері кіберзахисту, а також розглянуто підходи до моделювання, класифікації та реалізації сценаріїв кібератак. Отримані результати формують теоретичну основу для подальшого проєктування програмних систем симуляції кібератак.

.

РОЗДІЛ 2. МЕТОДИ ТА ІНСТРУМЕНТИ РОЗРОБКИ ПРОГРАМНИХ СИСТЕМ ДЛЯ СИМУЛЯЦІЇ КІБЕРАТАК

2.1 Архітектура та структура симуляційної моделі кіберполігону

До цієї роботи симулятор кібератак спочатку розроблявся з використанням програмного забезпечення ARENA. В результаті цієї розробки була створена модель з базовими можливостями редагування мережі та прийнятним рівнем можливостей специфікації атаки. Хоча ця модель ARENA довела, що концепція симуляції кібератак є правдоподібною, нові функції, бажані для моделі, такі як робота з XML-даними, виходять за межі можливостей ARENA. Бажання подолати такі обмеження з того часу спонукало до розробки незалежної моделі симуляції, яку можна відповідним чином налаштувати для реалізації функцій, які неможливі за допомогою симулятора ARENA. Ця нова модель включає налаштований інтерфейс, специфічний для розробки мережевих структур та сценаріїв атак, ряд параметрів введення та виведення, а також більш детальні функції визначення мереж та сценаріїв атак.

Модель була розроблена з використанням об'єктно-орієнтованої мови програмування Java. Java має багато бажаних характеристик для моделювання реалістичних середовищ. Java також є кросплатформною мовою програмування, яка вимагає лише JRE (середовище виконання Java) для запуску програми Java на будь-якій платформі. У решті цього розділу обговорюються основні цілі моделі, наводиться передумови для використовуваного середовища розробки та представлена структура елементів моделювання.

Наміри моделювання та огляд моделі

Загальною метою розробки цієї моделі симуляції кібератаки є створення застосунку, який може генерувати достовірні сповіщення системи виявлення вторгнень у віртуальній мережі, що представляє реальну приватну мережу. За допомогою симулятора кібератаки користувач може створити або завантажити певну топологію мережі, вказати вразливості мережі, створити та запустити

сценарії атаки, а також переглянути дані сповіщень датчиків [25]. Для функціональності симулятора необхідно кілька вхідних та вихідних даних. Діаграма, що зображує типи вхідних та вихідних даних, показана на рисунку 2.1.

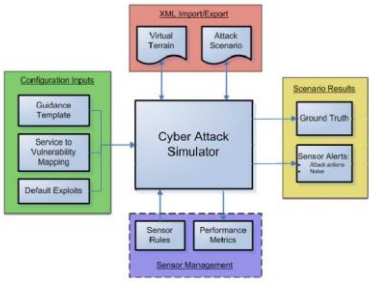

Рисунок 2.1 - Функціональність симулятора кібератаки

Як показано на рисунку 2.1, симулятор складається з трьох основних категорій вхідних та вихідних даних: вхідні дані конфігурації, імпорт та експорт XML, а також результати сценаріїв. Також показано додаткову програму для керування датчиками.

Вхідні дані конфігурації містять дані, які завантажуються з файлів під час відкриття симулятора. Файл шаблону інструкцій – це орієнтований граф, який вказує, яку послідовність етапів можна використовувати в атаці та які категорії експлойтів включені в кожен етап. Ця інформація використовується під час генерації атак на основі набору параметрів. Інший файл r містить зіставлення сервісів з вразливістю, яке зіставляє сервіс машини з набором вразливостей за допомогою ідентифікаторів сервісів та ідентифікаторів вразливостей [26].

Це фактично вказує на те, які експлойти можна виконати на машині, на якій запущено певну службу. Файл експлойтів за замовчуванням завантажує базу даних відомих експлойтів (також званих доступними діями), які симулятор може вибрати та відфільтрувати. Ця база даних дій використовується симулятором під час створення окремих кроків атаки, а також під час створення шуму (або хибнопозитивних результатів), що виникає під час сценарію атаки.

Імпорт та експорт XML дозволяють створювати або інтерпретувати структуровані XML-документи симулятором. Ці документи також можуть

створювати та інтерпретуватися іншими програмами, забезпечуючи засоби взаємодії симулятора з цими програмами [27]. Документ, який зазвичай використовується інструментами об'єднання інформації під час розробки, - це XML-документ «Віртуальний ландшафт». Цей документ зображує детальну структуру мережі, незалежно від того, чи є мережа реальною, чи віртуальною. На рисунку 2.2 показано, як документ віртуального ландшафту використовується різними програмами.

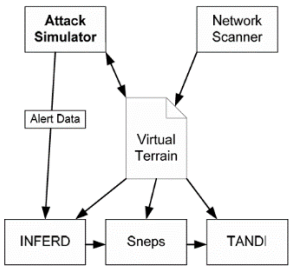

Рисунок 2.2 - Інтеграція віртуального ландшафту

Симулятор кібератаки може, зокрема, зчитувати модель мережі з віртуального ландшафту або створювати документ віртуального ландшафту з змодельованої мережі. Таким чином, програми, які використовують дані сповіщень, згенеровані симулятором, також можуть отримувати всю структуру мережі, яка використовується для генерації даних. Крім того, симулятор може створювати або зчитувати додатковий XML-документ, що зображує набір атак у сценарії атаки.

Результати сценарію включають дані, що генеруються під час виконання сценарію атаки та виводяться у текстові файли. Файл наземних даних містить усі дії на основі атаки, що відбуваються, а також такі деталі, як атака, до якої належала дія, джерело та місце призначення атаки, а також успіх або невдача дії. Набір файлів сповіщень датчиків містить усі сповіщення IDS, згенеровані датчиками, розміщеними в мережевій моделі, з окремим файлом для кожного датчика. Ці сповіщення IDS відповідають як діям на основі атаки, так і діям на основі шуму.

Нарешті, додаткова функція керування датчиками містить відповідну функціональність для визначення правил обробки згенерованих сповіщень датчиками. Моделюється черга Fusion Engine, і сповіщення надсилаються до цієї черги, як визначено правилами датчиків [28]. Метрики продуктивності генеруються для порівняння правил датчиків.

Перш ніж заглиблюватися в деталі структури програми, читачеві необхідно мати базове розуміння об'єктно-орієнтованого підходу Java. У цьому розділі описано деякі основні функції та переваги мови програмування Java. Ті, хто знайомий з базовою структурою Java (або подібної об'єктно-орієнтованої мови), можуть перейти до наступного розділу.

Мова Java реалізована як об'єктно-орієнтована мова програмування. Основний підхід в об'єктно-орієнтованому програмуванні полягає у представленні реальних об'єктів, таких як люди чи машини, як класів програмування з унікальними атрибутами та функціями (або методами), які ці об'єкти можуть надавати. Моделювання загалом намагається забезпечити точну модель певної реальної ситуації, якою можна маніпулювати та вивчати для оцінки існуючих або запропонованих систем [29]. Тому використання об'єктно-орієнтованого програмування при розробці структури моделювання має багато переваг, враховуючи, що моделі моделювання, як правило, представляють взаємодію між реальними об'єктами. Наприклад, у випадку симулятора кібератаки, клас Sensor використовується для представлення реального датчика IDS. Кожен датчик, включений до змодельованої мережі, представляє один екземпляр цього класу. Потім екземпляру датчика надаються унікальні атрибути, які включають розташування, тип, генеровані сповіщення тощо. Клас також включає методи, які може використовувати кожен екземпляр, такі як додавання нового сповіщення до списку сповіщень датчика або виведення всіх генерованих сповіщень у текстовий файл.

Окрім використання класів для представлення об'єктів, Java має інші ключові функції, які допомагають забезпечити модульність та можливість повторного використання, щоб зробити їх простими та гнучкими мовами програмування [30]. Ці функції включають успадкування, інкапсуляцію та поліморфізм.

Успадкування дозволяє розглядати клас або об'єкт як дочірній елемент іншого класу. Усі атрибути та методи, пов'язані з батьківським класом, успадковуються в дочірніх класах. Крім того, дочірні класи можуть оголошувати нові класи та атрибути, які роблять їх унікальними від батьківського. Також дочірній клас може перевантажувати методи, що використовуються в батьківських класах, щоб забезпечити іншу функціональність. Повертаючись до прикладу класу датчиків, два інших класи використовуються для конкретного представлення мережевих датчиків та датчиків на базі хоста. Оскільки ці класи є дещо більш специфічними версіями класу датчиків, вони використовують властивість успадкування, щоб отримати всі ті ж атрибути та методи, що й загальний датчик.

Інкапсуляція дозволяє за потреби захистити атрибути та методи, специфічні для класу, від інших класів. Роблячи атрибути та методи приватними таким чином, зовнішні класи повинні отримувати доступ до атрибутів та методів через певний інтерфейс до класу. Цей інтерфейс має форму публічних методів, де програміст може чітко визначити та обмежити, як можна переглядати та змінювати атрибути класу, а також як можна використовувати методи. Використання пакетів у Java – це ще одна форма інкапсуляції. Пакети дозволяють групувати кілька пов'язаних класів з меншими обмеженнями на взаємодію між класами.

Поліморфізм дозволяє кільком класам використовувати однакову поведінку, реалізуючи спільні методи. Це також включає можливість використання різних конструкторів для ініціалізації об'єктів (Joines and Roberts, 1998). Наприклад, у симуляторі кібератаки клас Machine використовує інший конструктор для визначення одного хоста машина, ніж клас використовує для визначення машини, що належить до кластера хостів (або підмережі).

Таким чином, можливість повторного використання та розширення об'єктів за допомогою описаних раніше функцій є центральною характеристикою та перевагою розробки фреймворку моделювання на Java. Крім того, оскільки об'єкти є модульними, інформація, пов'язана з кожним екземпляром об'єкта, зберігається лише в одному місці, з посиланнями на розташування об'єкта за потреби. Крім того, логіку моделювання моделювання можна належним чином розділити

між класами. Таке делегування функцій моделювання забезпечує більш організовану структуру, ніж це було б можливо за допомогою процедурного моделювання.

2.2 Програмні компоненти та структура пакетів системи симуляції

Класи симулятора організовані в шість пакетів на основі взаємодії класів. До них належать пакет симуляції, мережевий пакет, пакет атаки, візуальний пакет, пакет інтерфейсу віртуального ландшафту (VT) та пакет керування датчиками. Діаграма на рисунку 2.3 ілюструє цю структуру та перелічує деякі основні типи класів та/або функції кожного пакета.

Пакет моделювання містить класи для підтримки та моделювання за допомогою впорядкованого списку подій. Пакет також містить класи для генерації випадкових чисел [31]. Пакет network містить усі класи, що використовуються для визначення топології мережі та атрибутів пристроїв. Також підключення мережевих пристроїв реалізовано за допомогою методів усередині класів.

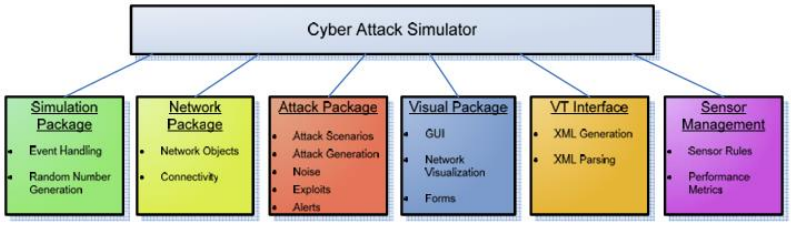

Рисунок 2.3 - Структура пакета симулятора кібератаки

Пакет атаки містить класи, необхідні для створення сценаріїв атаки, які складаються з серії атак на мережу разом із мережевим шумом. Пакет містить класи та методи, що використовуються для автоматичної генерації серії атак, коли вказані лише параметри атаки. Також пакет атаки містить класи, які діють як база даних для доступних дій атаки (або експлойтів) та сповіщень датчиків.

Пакет візуальних ефектів містить класи, що використовуються для забезпечення інтерфейсу користувача симулятора. Ці класи допомагають створити

графічний інтерфейс користувача (GUI) програми, візуалізацію мережі, форми введення даних та відображення результатів.

Пакет інтерфейсу VT містить чотири класи, що використовуються для розбору та створення XML-документів з даними віртуальної місцевості та даними атаки. Два з цих класів є статичними класами, які служать лише для надання серії змінних класів для представлення елементів та атрибутів XML. Решта два класи використовуються для читання та запису XML-документів, один клас присвячений даним віртуальної місцевості, а інший – даним атаки. Ці два класи використовують пакет JDOM, який є окремим загальнодоступним пакетом, що використовується для запису та інтерпретації XML-документів, коли надається зі структурою документа. Пакет JDOM не входить до базового комплекту розробки Java (JDK).

Пакет керування датчиками містить класи для визначення та реалізації набору правил роботи з датчиками. Пакет також містить клас Fusion Engine та спеціальні класи датчиків, що успадковуються від класів датчиків, визначених у пакеті network. Більше інформації щодо структури пакета керування датчиками можна отримати.

Пакет моделювання

Одним з ключових компонентів моделі є базова функціональність моделювання. Ця функціональність забезпечується за допомогою об'єктів та методів, які зазвичай зустрічаються в об'єктно-орієнтованому фреймворку моделювання [32]. У моделі пакет моделювання охоплює набір спеціально створених класів, які емулюють типову роботу фреймворку моделювання. На рисунку 2.4 показано структуру класів пакета моделювання у вигляді спрощеної діаграми UML.

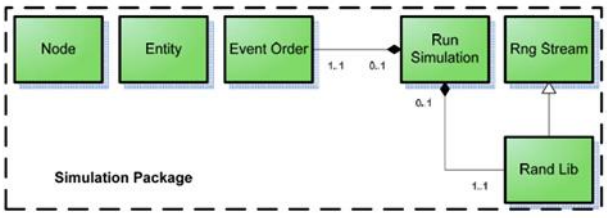

Рисунок 2.4 - Структура класів пакета моделювання

Структура, використана на рисунку 2.4, зображує позначені блоки як класи, тоді як лінії між двома блоками представляють зв'язок між класами. Спрямована стрілка вказує на успадкування; наприклад, клас Rand Lib успадковується від класу Rng Stream . З'єднання у формі ромба вказує на композицію; наприклад, екземпляр класу Event Order розглядається як атрибут класу Run Simulation. У таблиці 2.1 наведено детальну інформацію щодо кожного класу в пакеті.

Таблиця 2.1 -

Деталі класу пакета моделювання

Назва класу (Class Name)	**Тип (Type)**	**Опис (Description)**
Node (Вузол)	Абстрактний	Представляє фізичне місце розташування, куди сутності входять і звідки виходять. Цей клас не інстанціюється безпосередньо, а слугує суперкласом для класів у пакеті Network, що представляють мережеві пристрої.
Entity (Сутність)	Абстрактний	Представляє об'єкти, що переміщуються між вузлами. Клас також містить атрибути, які можуть бути змінені або використані вузлами. Не інстанціюється безпосередньо, а виступає суперкласом для специфічних класів у пакеті Attack, що представляють мережевий трафік.
Event Order (Порядок подій)	Конкретний	Клас порядку подій — це по суті список, який відстежує події, що мають відбутися в симуляції, у хронологічному порядку. Екземпляр цього класу розглядається як атрибут класу виконання симуляції (run simulation).
Run Simulation (Виконання симуляції)	Конкретний	Виконує дискретно-подієве моделювання сценарію атаки, вилучаючи події з класу порядку подій (та додаючи їх туди). Також відстежує час моделювання, який базується на часі поточної події.
Rng Stream (Потік ГПВЧ)	Конкретний	Клас потоку ГПВЧ (генератора псевдовипадкових чисел) забезпечує функціональність для створення випадкових чисел за умови надання початкового значення (seed). Це загальнодоступний клас, створений L'Ecuyer et al. (2002).
Rand Lib (Бібліотека випадковостей)	Конкретний	Клас rand lib успадковує властивості rng stream і надає додаткову функціональність, що дозволяє визначати кілька потоків. Також забезпечує можливість вибору чисел із певного розподілу (наприклад, рівномірного або експоненціального).

Шість класів, що входять до пакету моделювання, забезпечують основу для запуску моделювання дискретних подій у симуляторі кібератаки; однак реалізація

Ці елементи моделювання суттєво інтегровані як з мережевим, так і з пакетом для обробки атак [33]. Тому, перш ніж надавати більш детальну інформацію про моделювання, необхідно чітко розуміти обидва ці пакети. Розділ цієї роботи (методологія моделювання атаки) містить більш детальний опис функціональності елементів моделювання та того, як вони інтегровані з іншими пакетами.

Мережевий пакет

Модель включає набір класів, що використовуються для визначення віртуальної комп'ютерної мережі. Пакет network охоплює цей набір класів, які варіюються від фізичних мережевих пристроїв до програмних налаштувань на цих пристроях.

На рисунку 2.5 показано структуру класів пакета network, наведено детальну інформацію про кожен клас, що входить до пакета.

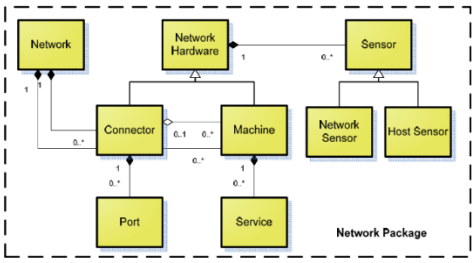

Рисунок 2.5 - Структура класу мережевого пакета

Як згадувалося раніше, абстрактний клас Node в пакеті моделювання використовується як надклас фізичних мережевих компонентів. Це досягається завдяки класу мережевого обладнання, що успадковується від класу node. Таким чином, як з'єднувачі, так і об'єкти машин представляють вузли, між якими можуть переміщатися сутності. Це є репрезентативним для реальної мережі, оскільки пакети інформації переміщуються визначеними шляхами між мережевими пристроями. Крім того, хоча немає конкретного об'єкта підмережі (або кластера хостів), підмережа визначається шляхом надання серії об'єктів машин з однаковим ідентифікатором групи [34]. Таким чином, кожен член підмережі все ще

Функціонує незалежно, але може керуватися колективно

Пакет атаки

З огляду на стаціонарні аспекти моделі кібератаки, визначені в мережевому пакеті, пакет атаки призначений для включення «мобільних» компонентів моделі (або принаймні мобільних щодо руху інформації). Пакет атаки, по суті, включає класи для моделювання атак, які можуть проходити через комп'ютерну мережу, а також іншого трафіку, не пов'язаного з атакою. На рисунку 2.6 показано структуру класів пакета атаки.

Пунктирні стрілки на рисунку 2.6 представляють зв'язок залежності. У цьому зв'язку клас залежить від іншого класу або за його методи, або за атрибути; однак ні успадкування, ні композиція не використовуються. Наприклад, клас Attack отримує доступ до атрибутів класу в абстрактному класі Guidance Template, навіть якщо ці два класи фізично жодним чином не пов'язані між собою.

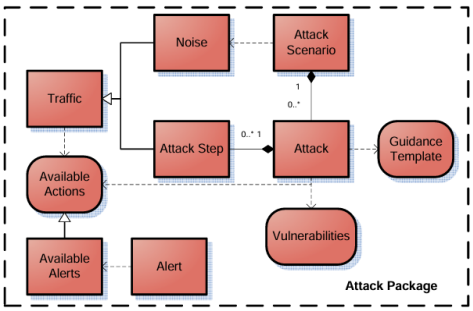

Рисунок 2.6 - Структура класу пакета атаки

Крім того, клас трафіку успадковується від абстрактного класу сутності , визначеного в пакеті моделювання. Визначення трафіку як підкласу класу сутності дозволяє маршрутизувати трафік між вузлами в структурі мережі так само, як реальний мережевий трафік маршрутизував бись через пристрої в реальній мережі. Надається додаткова методологія для заборони або дозволу маршрутизації трафіку між двома вузлами та для генерації сповіщення, якщо на вузлі присутній датчик. Обговорюється методологія, яка використовується для управління

потоком мережевого трафіку, а також методологія, пов'язана зі створенням атак та сценаріїв атак.

Важливим аспектом використання програми є графічний інтерфейс користувача (GUI). Графічний інтерфейс симулятора кібератаки забезпечується через численні класи, що містяться у візуальному пакеті. На рисунку 2.7 показано деякі основні класи (та інтерфейси), що входять до складу цього пакету.

Маленькі кола, прикріплені до класів на рисунку 2.7, представляють собою користувацькі інтерфейси, реалізовані цими класами. Інтерфейс у Java реагує на певні дії (у цьому випадку дії користувача), щоб викликати методи в іншому класі [35]. Наприклад, монітор миші виявляє клацання та рух миші користувачем. Візуальний пакет також містить понад двадцять інших форм, не показаних на рисунку. Ці форми є або підкласами базової мережевої форми , або підкласами якоїсь іншої форми, але всі форми успадковують властивості та методи базової мережевої форми. Ці додаткові форми використовуються як для введення, так і для перегляду даних для симулятора.

Після запуску моделі користувачеві спочатку надається екземпляр класу програми кібератаки разом із усіма завантаженими стандартними даними. Цей інтерфейс дозволяє користувачеві завантажувати мережі, створювати мережі або змінювати налаштування програми [36]. Додаткові меню та опції стають доступними після активного відображення моделі мережі. На рисунку 2.8 показано знімок екрана програми з двома прикладами відображення мереж. На рисунку також показано деякі ключові функції програми (червоні прямокутники/стрілки).

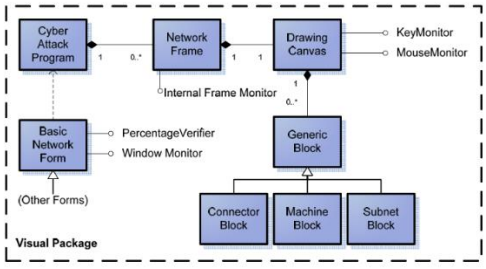

Рисунок 2.7 - Структура класу візуального пакету.

На рисунку 2.8 для мережі, показаної на передньому плані, намальовані об'єкти представляють візуальні компоненти мережі. Маршрутизатор 1 та маршрутизатор 2 є екземплярами класу блоків з'єднувачів, які візуально представляють екземпляри класу з'єднувачів (у пакеті мережі).

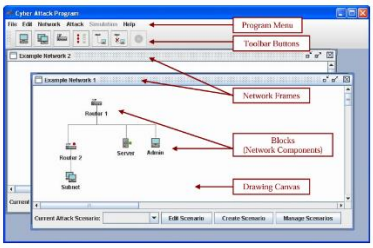

Рисунок 2.8 - Знімок екрана програми кібератаки.

Сервер та Адміністратор є екземплярами класу блоків машин, які представляють об'єкти машин. Підмережа є екземпляром класу блоків підмережі та представляє набір об'єктів машин. Клас мережевого полотна також надає методологію для малювання з'єднання між об'єктами в мережі, які фізично пов'язані. На рисунку показано, що маршрутизатор 2, сервер і адміністратор підключені до маршрутизатора 1, а підмережа - до маршрутизатора 2.

Блоки на полотні можна переміщувати, а подвійне клацання на певному блоці викличе форму для редагування об'єкта, який представляє цей блок. Додаткові блоки, з'єднання та навіть сценарії атаки можна налаштувати за допомогою відповідних кнопок, розташованих у фреймі мережі, панелі інструментів програми та меню програми. Крім того, програма надає інтерфейс форм, що використовуються як для запуску моделювання, так і для відображення результатів після створення мережі та сценарію атаки для мережі.

2.3 Методології моделювання кібератак та емуляції поведінки зловмисника

Після визначення детальної структури мережі за допомогою компонентів мережевого пакета, мережу можна використовувати як основу як для

налаштування, так і для моделювання серії кібератак. Пакет атак надає засоби для детального опису та моделювання атак у застосунку. Пакет також надає основну частину функціональності, необхідної для створення сповіщень датчиків. У цьому розділі розглядається методологія, яка використовується для створення мережевих атак та сповіщень датчиків, що є репрезентативними для реальних кібератак та пов'язаних з ними сповіщень IDS.

Коли сценарій атаки повністю визначено, програма може перейти до запуску моделювання дискретних подій атак, що відбуваються через мережу, та датчиків, які генерують сповіщення. Цей процес моделювання відбувається шляхом обробки подій. Також процес моделювання подій надає можливість генерувати XML-документ, що містить деталі атаки для всіх атак у сценарії.

Моделювання трафіку

У реальній комп'ютерній мережі мережевий трафік являє собою всі пакети, що переміщуються між пристроями в мережі. Моделювання всього можливого мережевого трафіку є трудомістким процесом, який також суттєво знижує продуктивність моделювання. Крім того, моделювання цього трафіку не додає жодної додаткової цінності до моделювання. Тому модель включає лише мережевий трафік, який бере участь у розвитку атаки або процесах виявлення вторгнень.

Модель розглядає три типи мережевого трафіку: трафік, пов'язаний з кроком атаки (або дією), що відповідає визначенню сповіщення, трафік, пов'язаний з кроком атаки, що не відповідає визначенню сповіщення, та трафік, який не пов'язаний з кроком атаки, але все ще має визначення сповіщення. Останній тип стосується шуму (або хибнопозитивних результатів), що являють собою цілком нормальну активність, яку можна помилково прийняти за шкідливу. Трафік, який не пов'язаний ні з кроком атаки, ні з визначенням сповіщення, не моделюється, оскільки такий трафік не впливатиме на атаки або датчики IDS [37].

Модель використовує об'єкти трафіку для визначення кожної сутності трафіку, яка переміщується між мережевими пристроями. Однак ці об'єкти не представляють окремі пакети. Натомість, об'єкти представляють набір пакетів,

необхідних для виконання певної дії. Представлення трафіку таким чином зменшує обсяг інформації, яку необхідно маршрутизувати між вузлами мережевої моделі. Основні атрибути, визначені в класі трафіку, включають вихідну машину, цільову машину, індекс дії, категорію дії та час створення. Клас трафіку також включає набір змінних класу, які вказують на можливі первинні та вторинні категорії для дій.

Вихідна машина являє собою машину, яка генерує трафік, і вказує на вузол у мережевій моделі, на якому спочатку розташований об'єкт трафіку під час запуску симуляції [38]. Цільова машина являє собою машину, для якої призначена дія трафіку. Після досягнення цього цільового вузла трафік видаляється з симуляції. Процес, який використовується для маршрутизації об'єкта трафіку між цими вузлами, описано в наступному розділі.

Індекс дії – це ціле число, яке посилається на дію, що виконується трафіком. Ця дія може бути зловмисним кроком в рамках атаки або просто звичайним шумом. Деталі дії містяться в класі доступних дій та посилаються на них індексом дії. Представлено детальну інформацію про дії, доступні в моделі, та пояснено, як використовується індекс дії. Крім того, об'єкт трафіку містить категорію дії, щоб вказати, з якою широкою категорією пов'язана дія.

Можливі первинні та вторинні категорії, перелічені у змінних класу трафіку, представляють систему класифікації, яка використовується для дій. Ця класифікація здійснюється за допомогою набору з п'яти первинних категорій, кожна з яких має підмножину вторинних категорій. Розподіл категорій показано на рисунку 2.9.

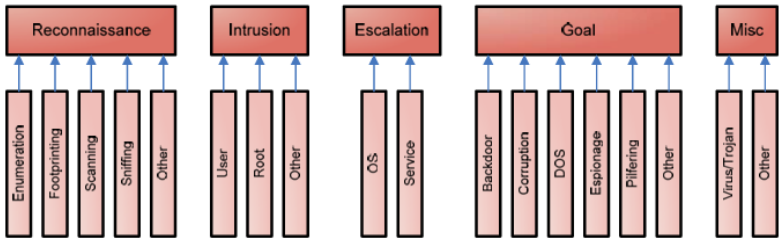

Рисунок 2.9 - Розподіл категорій дій.

Кожна виконана дія (і, отже, кожен об'єкт трафіку) вважається членом однієї з категорій, перелічених на цьому рисунку.

Хоча клас трафіку визначає достатній рівень деталізації для генерації мережевого трафіку, для фактичного створення об’єктів трафіку він спирається на два інших підкласи. Цими двома класами є клас етапу атаки та клас шуму, і кожен із них успадковує атрибути та методи, передбачені в класі трафіку.

Клас кроку атаки визначає трафік, пов'язаний з окремим кроком атаки. Для об'єкта кроку атаки дія, на яку посилається, представляє експлойт, виконаний на цільовій машині. Об'єкт кроку атаки також включає деякі додаткові атрибути, пов'язані із загальною атакою. Атрибут атаки вказує на об'єкт атаки, до якого належить крок [39]. Також атрибут номера кроку вказує, де відбувається крок у розвитку атаки. Нарешті, атрибут успіху – це логічне значення, яке вказує, чи був крок атаки успішно виконаний.

Клас шуму визначає трафік, пов'язаний з типовим нешкідливим мережевим шумом. Цей тип трафіку все одно генеруватиме сповіщення IDS, але дія, на яку посилається об'єкт шуму, представлятиме хибнопозитивний результат (коли дія насправді не відбулася або не відбулася як частина атаки). Клас шуму не визначає жодних додаткових атрибутів, але надає методологію для генерації випадкового шумового трафіку.

Доступні дії

Одним з ключових вхідних даних для моделі є база даних дій, доступних для використання при генерації як кроків атаки, так і мережевого шуму. Що стосується кроків атаки, то ці дії представляють собою потенційні експлойти, які можна використовувати під час атаки. Однак, що стосується шуму, то ці дії представляють собою потенційний набір хибнопозитивних результатів, з яких можна вибрати при генерації шумових об'єктів.

Клас доступних дій – це абстрактний клас, який спеціально визначає список дій, які можна використовувати. Цей клас містить масиви атрибутів для представлення атрибутів кожної дії. Для кожної окремої дії використовується унікальний індекс, що представляє позицію в масиві. Використовуючи цей підхід,

будь-який атрибут дії можна отримати за допомогою методів класу, якщо надано лише індекс. Додаток надає таблицю в меню налаштувань, де можна переглянути доступні дії за замовчуванням. Ця частина меню налаштувань показана на рисунку 2.10.

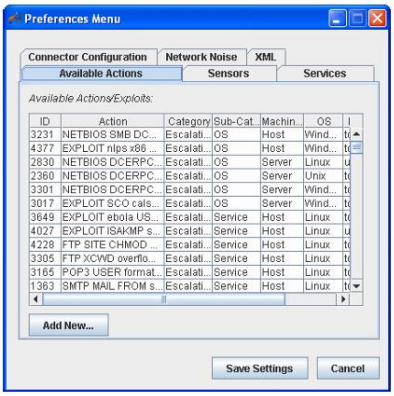

Рисунок 2.10 - Доступні дії, що відображаються в меню «Налаштування»

Масиви атрибутів, встановлені в класі доступних дій, включають дію, основну категорію, вторинну категорію, тип машини, операційну систему, ідентифікатор вразливості, протокол і номер порту. Ці масиви заповнюються текстовим файлом, який містить усі дії за замовчуванням, що використовуються в програмі. Цей текстовий файл завантажується на початку програми [40].

Атрибут action (дія) представляє фактично виконану дію. Цей атрибут можна вважати синонімом сигнатури, хоча він не представляє конкретний шістнадцятковий підпис пакета. Атрибут action використовується як назва дії та як атрибут, який сенсори IDS розглядають під час порівняння мережевого трафіку з базою даних сигнатур сповіщень.

Первинні та вторинні категорії класифікують дію та вказують на тип діяльності, що виконується. Ця інформація використовується під час фільтрації списку потенційних дій на основі потрібної категорії. Також ця інформація використовується під час розробки схеми розвитку атаки за допомогою етапів.

Тип машини, операційна система та ідентифікатор вразливості вказують на тип даних, необхідних на цільовій машині для успішного виконання дії.

Наприклад, залежно від типу машини та операційної системи, деякі дії можливі лише на сервері або на машині з Windows 2000. Дії, можливі на кількох типах машин або операційних системах, повинні мати запис у базі даних для кожного типу. Ідентифікатор вразливості представляє конкретну вразливість, яку може використати дія, і цей атрибут використовується для зіставлення служб машини з потенційними вразливостями, що виникають під час їх використання. Тому, коли на цільовій машині працює набір служб, на основі ідентифікаторів вразливостей можна отримати набір потенційно успішних дій.

Атрибути протоколу та номера порту вказують на альтернативний порт, який може використовувати дія. За замовчуванням дія використовуватиме один із сервісних портів сервісу, пов'язаного з дією. Однак для успішного виконання дії об'єктом трафіку потрібно, щоб принаймні один із цих сервісних портів був дозволений на всьому шляху, яким об'єкт трафіку рухається через мережу.

2.4 Висновок по розділу

У другому розділі було досліджено методи та інструменти розробки програмних систем для симуляції кібератак. Розглянуто архітектуру та структуру симуляційної моделі кіберполігону, що забезпечує відтворення реалістичних умов функціонування інформаційних систем. Проаналізовано програмні компоненти та структуру пакетів системи симуляції, які відповідають за взаємодію модулів, обробку подій та керування сценаріями атак. Також досліджено методології моделювання кібератак та емуляції поведінки зловмисника для створення ефективних тренувальних середовищ. У результаті визначено ефективні технологічні підходи до побудови програмних систем кібертренування.

РОЗДІЛ 3. РОЗРОБКА, ІНТЕГРАЦІЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Реалізація сценаріїв кібератак та тренувальних середовищ

Концепція сценарію атаки використовується в моделі для чіткого визначення типу активності, яка відбувається під час запуску симуляції. Ця активність включає як загрозливі атаки, так і типовий мережевий шум. Кожна змодельована мережа може містити набір сценаріїв атаки, визначених спеціально для мережі; однак симуляція обробляє лише один вибраний сценарій за раз. У мережевих фреймах передбачено спеціальну панель для керування сценаріями атаки мережі. Ця панель відображається на рисунку 3.1. За допомогою цієї панелі користувач може вибрати сценарій, створити сценарій, редагувати сценарій або керувати списком сценаріїв, включаючи видалення або копіювання сценарію.

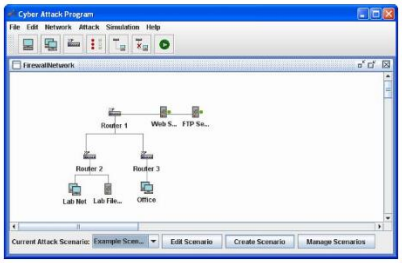

Рисунок 3.1 - Панель сценаріїв атаки на мережевому фреймі

Під час створення нового сценарію або редагування існуючого сценарію користувачеві надається форма, де можна змінити атрибути сценарію. Ця форма показана на рисунку 3.2. Кожен об'єкт сценарію атаки має набір атрибутів, що стосуються шуму, що генерується сценарієм, та набір атрибутів, що стосуються атак, включених до сценарію.

Шумовий трафік, що використовується у сценарії атаки, генерується під час симуляції. Цей процес суттєво зменшує обсяг інформації, яку потрібно зберігати у сценарії атаки. Генерація цього шуму керується набором параметрів

шуму, які є атрибутами сценарію атаки. Ці параметри включають розподіл (у відсотках) категорій дій, що використовуються в шумі, та швидкість, з якою генерується шум. Користувач може вказати, який відсоток (зі 100%) кожна з п'яти основних категорій дій буде враховувати у генерованому шумі. Поле розвідки, надане у формі, автоматично налаштується таким чином, щоб усі відсотки в сумі дорівнювали 100%. Атрибут рівня шуму (який у формі сценарію називається «сповіщення за годину») приблизно показує, скільки шумових об'єктів буде генеруватися за змодельовану годину. Ця швидкість розподілена експоненціально. Сценарій також складається з набору атак, які будуть виконані під час симуляції. Ці атаки визначаються перед виконанням симуляції, і сценарій атаки може містити стільки атак, скільки забажає користувач. Інтерфейс, наданий у формі сценарію атаки (рисунок 3.2), дозволяє додавати атаки до сценарію, редагувати або видаляти зі сценарію.

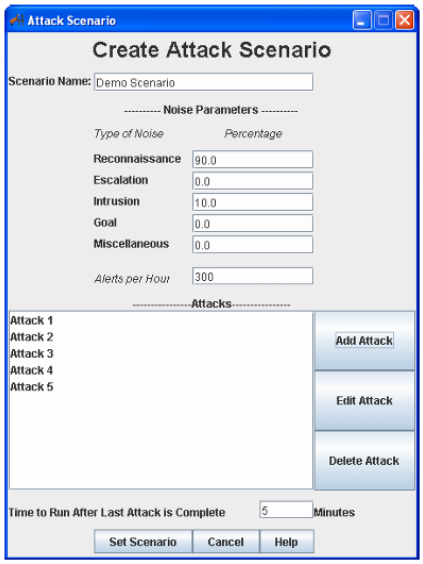

Рисунок 3.2 - Форма сценарію атаки

Крім того, сценарій атаки містив атрибут, який вказує, як довго (у змодельованому часі) має продовжуватися симуляція після завершення атак. Без цієї функції останній об'єкт трафіку завжди був би пов'язаний з кроком атаки. Таким чином, надання додаткового часу симуляції дозволяє генерувати більше шуму та уникає упередженості, коли симуляція завершується останнім кроком атаки.

Ще одним важливим атрибутом сценарію атаки, хоча й не показаним у формі, є генератор випадкових чисел. Сценарій забезпечується об'єктом rand lib, який використовується для надання всіх випадкових чисел, що використовуються під час ініціалізації атак сценарію. За бажанням, цей атрибут дозволяє використовувати той самий набір випадкових чисел щоразу, коли ініціалізуються атаки.

Атаки

Клас атаки використовується для визначення окремих (автономних) атак, що відбуваються як частина сценарію атаки. Атака вважається спрямованою на певну цільову машину з певною цільовою дією. Атака також охоплює набір кроків (або окремих експлойтів), що виконуються для досягнення кінцевої цільової дії. Простий приклад атаки зображено на рисунку 3.3.

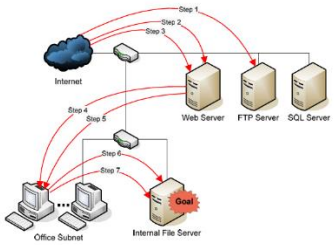

Рисунок 3.3 - Приклад розвитку атаки

У цій атаці кінцевою метою є встановлення бекдору на, здавалося б, безпечному внутрішньому файловому сервері мережі. Зовнішній хакер спочатку виконує розвідувальні дії для сканування серверів з прямим зовнішнім (інтернет) доступом. Потім хакер виконує крок вторгнення, щоб отримати доступ до веб-сервера. З веб-сервера хакер виконує дію вторгнення та дію ескалації, щоб отримати root-доступ до машини у внутрішній підмережі. Отримавши root-доступ, зловмисник може проникнути на файловий сервер і виконати останню дію бекдору.

Інтерфейс для додавання атаки до сценарію атаки пропонує два різні методи визначення атаки. Ці методи включають ручне визначення кроків атаки або автоматичну генерацію кроків атаки. Якщо користувач бажає ідентифікувати

кожну дію, що використовується в процесі атаки, тоді можна вибрати ручне налаштування атаки. Однак, якщо користувача цікавить лише кінцева мета, досягнута атакою, і його не дуже хвилюють точні кроки, що використовуються для досягнення мети, тоді можна вибрати автоматичне налаштування атаки.

Клас атаки надає атрибути та методи, що використовуються для створення детальних об'єктів атаки. Деякі з важливих атрибутів, визначених у класі атаки, включають тип атаки (ручна або автоматична), назву атаки, використану мережу, сценарій атаки, до якого включена атака, кінцеву цільову машину, кінцеву категорію мети, кроки атаки, час початку атаки, загальний час атаки, експоненціальний параметр, параметр ефективності, параметр прихованості та параметр навичок. Об'єкт атаки зберігає свої кроки як масив об'єктів кроків атаки. Крім того, атрибут загального часу атаки використовуватиметься лише в тому випадку, якщо час окремих кроків не визначено. Експоненціальний параметр – це логічний атрибут, який вказує, чи розподіляється загальний час атаки (якщо визначено) експоненціально навколо значення, чи використовується постійне значення. Три атрибути параметрів (ефективність, прихованість та навичок) використовуються при визначенні автоматично згенерованої атаки, і ці параметри будуть розглянуті.

Під час додавання ручної атаки до сценарію або редагування атаки, визначеної вручну, користувачеві відображається форма, яка зберігає кроки атаки у таблиці. Ця форма ручної атаки відображається на рисунку 3.4.

Рисунок 3.4 - Форма ручної атаки

Таблиця кроків відображає всі атрибути об'єктів кроків атаки, що використовуються в атаці. Ця форма також містить опції для позначення назви атаки, способу визначення часу атаки та часу початку атаки.

Додавання нового кроку або редагування існуючого кроку призведе до створення нової форми відображається, що містить поля для визначення атрибутів кроку атаки. Ця форма кроку атаки показана на рисунку 3.5. Форма, по суті, дозволяє визначити шлях трафіку, використану дію та час, відведений для атаки.

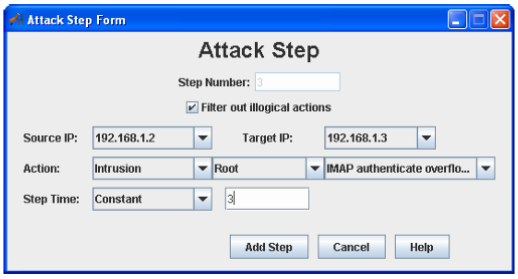

Рисунок 3.5 - Форма кроку атаки

У формі кроку атаки поля «Вихідна IP-адреса» та «Цільова IP-адреса» вказують IP-адресу вихідного та цільового комп'ютерів для трафіку кроку атаки відповідно. На основі вибраної вихідної IP-адреси список доступних IP-адрес у полі «Цільова IP-адреса» буде відфільтровано, щоб відображати лише ті комп'ютери, до яких можна отримати доступ з комп'ютера, представленого вихідною IP-адресою. Дія вибирається шляхом зазначення основної та додаткової категорій дій (перші два поля після мітки «Дія»), а потім вибору з доступних дій, що відповідають цим категоріям. Категорія додаткової дії не є обов'язковою, але допомагає звузити список доступних дій. Якщо встановлено прапорець для фільтрації нелогічних дій, вибрана цільова машина (шляхом посилання на цільову IP-адресу) виступає ще одним критерієм для фільтрації списку дій, які можна виконати. Розділ пояснить методологію надання лише логічних дій (зокрема, експлойтів) для кроків атаки. Нарешті, час, відведений для кроку, може бути заданий як постійне або експоненціально розподілене значення. Цей час кроку вказує, скільки змодельованого часу пройде, перш ніж можна буде виконати нас-

тупний крок.

Налаштування автоматично згенерованої атаки використовує іншу форму, яка не містить інформації про конкретні кроки. Форма автоматичної атаки показана на рисунку 3.6. Форма містить ті ж поля, що й форма ручної атаки, для налаштування назви атаки та інформації на основі часу. Однак форма містить набір параметрів атаки та варіантів цілей як заміну окремих кроків атаки. Параметри ефективності, скритності та майстерності – це значення, які можна вказати зі значеннями від 0 до 1. Значення кожного параметра буде обговорено, але основна передумова цих параметрів полягає в широкому визначенні характеристик кроків атаки, які будуть згенеровані. Поле «Ціль» вказує на кінцеву цільову машину атаки (вибрану за IP), а тип цілі вказує на те, яка категорія вторинної цілі має бути використана для кінцевої дії, що виконується в атаці.

Рисунок 3.6 - Форма автоматичної атаки

Усі параметри, визначені у формі автоматичної атаки, надають достатньо інформації для того, щоб модель автоматично генерувала кроки атаки. У підрозділі буде представлено методологію, яка використовується моделлю для генерації кроків атаки з параметрів автоматичної атаки.

Логічні експлойти

Під час визначення кроків атаки, що відбуваються під час атаки, доступні опції повинні відображати, який тип активності фактично можливий у змодельованій мережі. Характеристики мережевого пристрою, такі як запущені служби та брандмауери, присутні на конекторах, обмежуватимуть тип трафіку (і, отже,

кроки атаки), який може логічно виникати. Тому як інтерфейс для визначення кроків атаки, так і процес автоматичної генерації кроків атаки включають методологію, необхідну для того, щоб дії, що виконуються під час атаки, являли собою логічні експлойти.

У формі кроку атаки ця логіка забезпечується шляхом посилання як на атрибути цільової машини, так і на атрибути конекторів, що знаходяться вздовж маршруту від вихідної машини до цільової машини. Щодо цільової машини, атрибутами, що цікавлять, є тип машини, операційна система та запущені служби. Список доступних дій, наданих у формі, буде відфільтровано, щоб включити лише ті дії, де тип машини відповідає типу цільової машини, операційна система відповідає операційній системі цілі, а ідентифікатор вразливості включено до зіставлення ідентифікаторів служб з ідентифікаторами вразливостей на цільовій машині. Щодо конекторів вздовж маршруту від вихідної до цільової машини, атрибутом, що цікавить, є список дозволів брандмауера. Доступні дії, перелічені у формі, будуть додатково відфільтровані, щоб включити лише ті дії, які не будуть заблоковані дозволами брандмауера. Ця фільтрація виконується шляхом перевірки портів, що використовуються службою, пов'язаною з кожною дією, щоб переконатися, що принаймні один із цих портів на основі служб дозволено вздовж усього маршруту від джерела до цільової машини. Якщо пов’язана служба не містить жодних портів (наприклад , служба «Загальні»), тоді інформація про альтернативний порт дії, надана через клас доступних дій, перевіряється на відповідність дозволам брандмауера.

Хоча фільтрація нелогічних дій допомагає визначити кроки атаки, які будуть успішними, цю опцію також можна знехтувати, щоб відобразити всі дії . Користувач також може захотіти, щоб певний крок атаки включав дію, яка зазнає невдачі.

Процес генерування шуму також використовує подібний підхід для фільтрації нелогічних дій. Цільова машина об'єкта шуму та дозволи брандмауера з'єднувачів уздовж маршруту від джерела до цільової машини забезпечать засоби для фільтрації дій, які можна вибрати під час процесу генерування шуму.

Параметри автоатаки

Параметри, що входять до визначення автоматичної атаки, використовуються для керування генерацією кроків атаки. Ці параметри включають ефективність, прихованість та майстерність, і ці параметри призначені для моделювання поведінки хакера, який виконує атаку.

Ефективність стосується прямості атаки. Цей параметр по суті вказує на те, наскільки добре хакер може просуватися безпосередньо до кінцевої цілі. Доступні значення для параметра – це десяткові числа в діапазоні від 0 до 1, де 0 представляє дуже неефективну атаку, а 1 – високоефективну пряму атаку. На рисунку 7.9 зображено два різні типи атак в одній мережі. Атака, показана ліворуч на рисунку, являє собою високоефективну атаку, тоді як атака, показана праворуч, являє собою дещо неефективну атаку. Відповідні значення параметра ефективності для цих атак можуть бути 0,9 для атаки ліворуч та 0,4 для атаки праворуч.

Значення ефективності визначається під час процесу генерації автоматичної атаки при спробі просування вниз по мережі до цілі. Випадкове число (від 0 до 1) вибірково вибирається з об'єкта rand lib сценарію атаки та порівнюється зі значенням ефективності. Якщо ефективність перевищує вибіркове значення, атака переміститься на нижчий рівень мережі або на кінцеву цільову машину, якщо машина знаходиться на поточному рівні. В іншому випадку атака переміститься на іншу машину того ж рівня.

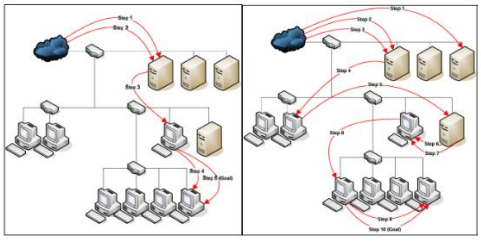

Рисунок 3.7 - Приклад ефективності

Прихованість стосується того, наскільки добре зловмисник уникає проміжних кроків цілі. Виконання багатьох кроків цілі на додаток до кінцевої цілі

значно полегшує виявлення атаки мережевими адміністраторами та виявлення системами виявлення вторгнення. Тому мінімізація кількості додаткових кроків цілі допомагає сформувати приховану атаку, яка краще прихована в мережевому трафіку. Параметр прихованості також приймає десяткове значення від 0 до 1, де 0 представляє дуже виявлювану атаку, яка виконує цільові кроки на кожній машині, що зустрічається, а 1 представляє приховану атаку, яка виконує лише кінцеву ціль. Параметр прихованості також перевіряється на вибірку випадкового числа в певні моменти під час автоматичної генерації атаки, щоб визначити категорію кроку атаки.

Навички визначають, наскільки успішно хакер виконує відповідні кроки протягом усього процесу атаки. Вправний хакер отримає достатньо інформації про мережу , щоб виконати дії, які будуть успішними. Однак некваліфікований хакер зазвичай пробує випадковий набір інструментів атаки, використовуючи дії, які можуть бути успішними або ні, залежно від деталей та безпеки мережі. Параметр навичок відображає ймовірність невдачі атаки в певний момент (коли кінцева мета не буде досягнута). Значення параметра може бути десятковим числом у діапазоні від 0 до 1, де 0 означає атаку, яка, ймовірно, зазнає невдачі, а 1 – атаку, яка, ймовірно, буде успішною. Однак значення 1 не обов'язково означає, що атака буде успішною. Мережа може бути налаштована таким чином, що жоден експлойт не буде успішним проти певної машини. У цій незвичайній ситуації навіть ідеально кваліфікований хакер не зможе виконати жодного кроку на такій машині. У процесі автоматичної генерації атаки параметр навички (у порівнянні з вибірковим значенням) по суті визначає різницю між вибіркою лише з логічних експлойтів (дій) та вибіркою з усіх доступних дій під час вибору конкретної дії кроку атаки.

Шаблон інструкції

Одним з основних вхідних даних для процесу автоматичної генерації атаки є шаблон інструкцій. Шаблон інструкцій являє собою графічний шаблон, розроблений, щоб вказати, що потрібно для визначення прогресу атаки. Текстовий файл використовується для зберігання як пріоритету етапів, та категорії дій,

доступні на кожному етапі. Поточні (за замовчуванням) категорії дій для кожного етапу наведено в таблиці 3.1 (із зазначенням як основної, так і додаткової категорії).

Таблиця 3.1

Категорії дій на етапах шаблону інструкцій

Етап (Stage)	**Категорії дій (Action Categories)**
Етап 0	Вторгнення/Інше, Розвідка/Перерахування, Розвідка/Збір інформації (Footprinting), Різне/Інше
Етап 1	Вторгнення/Користувач, Вторгнення/Інше, Різне/Інше
Етап 2	Підвищення привілеїв/Служба, Підвищення привілеїв/Інше
Етап 3	Вторгнення/Root (Суперкористувач)
Етап 4	Ціль/DoS (Відмова в обслуговуванні), Ціль/Бекдор, Ціль/Розкрадання даних
Етап 5	Вторгнення/Інше, Розвідка/Перерахування, Розвідка/Збір інформації, Різне/Інше, Сканування
Етап 6	Вторгнення/Користувач, Вторгнення/Інше, Різне/Інше
Етап 7	Підвищення привілеїв/Служба, Підвищення привілеїв/Інше

Етапи від 0 до 4 представляють прогрес атаки, що використовується для виконання дій на машині із зовнішнім доступом, тоді як етапи від 5 до 9 представляють прогрес атаки, що використовується для виконання дій на машинах у внутрішній частині мережі. Оскільки ці два завдання мають багато спільного, етапи від 0 до 4 представляють майже ті ж категорії дій, що й етапи від 5 до 9 відповідно.

Клас шаблону інструкцій діє як абстрактний клас, який зберігає як список категорій дій для кожного етапу, так і пріоритет, необхідний для досягнення кожного етапу. Будь-який з цих атрибутів можна отримати з класу шаблону інструкцій, використовуючи значення (індекс) етапу. Оскільки клас шаблону інструкцій заповнюється за допомогою вхідного файлу, модель має певну гнучкість, оскільки користувач може змінювати або пріоритет етапів, або категорії дій, щоб забезпечити різні (і, можливо, більш конкретні) послідовності атак.

Під час виконання автоматичної генерації атаки етап 0 буде використано як перший крок атаки, оскільки жодні інші етапи недоступні до завершення

цього етапу. Після виконання дії з цього етапу доступні етапи з 1 по 4. Однак етап 4 буде використано лише за умови, що параметр прихованості дозволяє. Як тільки кроки атаки почнуть націлюватися на внутрішню машину, процес генерації атаки розпочнеться з використанням етапів з 5 по 9. Етап 9, як і етап 4, буде використано лише за умови, що параметр прихованості дозволяє; однак кінцева мета атаки також використовуватиме етап 9.

Методологія автоатаки. Клас атаки включає набір методів, що використовуються для автоматичної генерації кроків атаки, що використовуються в атаці. Ці методи викликаються класом сценарію атаки безпосередньо перед запуском симуляції. Клас сценарію атаки послідовно переглядатиме кожну атаку, включену до сценарію, щоб переконатися, що кроки атаки визначені. Якщо атака не має визначеного набору кроків, будуть викликані методи автоматичної генерації атаки. Симулятор надає інтерфейс, який відображає користувачеві детальну інформацію про цей процес. Цей інтерфейс показано на рисунку 3.8 (відразу після ініціалізації атак). Користувач має можливість використовувати загальний набір випадкових чисел, які генеруватимуть однакові атаки щоразу, або певне початкове значення випадкового числа, що по суті призведе до різних атак.

Перш ніж можна буде сформувати етапи атаки, необхідно виконати кілька попередніх кроків. По-перше, слід перевірити мережу, щоб переконатися, що якась точка має доступ до зовнішньої мережі (Інтернету). Далі об’єкт rand lib сценарію атаки скидає потік генерації випадкових чисел із тим самим початковим насінням (для загального випадкового числа) або із заданим насінням. Цей потік скидається лише один раз (перед обробкою першої атаки).

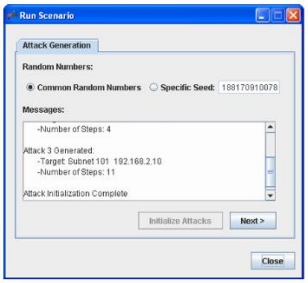

Рисунок 3.8 - Інтерфейс автоматичного формування атак

Після скидання потоку випадкових чисел методологія проходить кожну атаку окремо. Кінцева цільова машина розташована в мережі, і створюється критичний шлях від точки зовнішнього доступу до цієї цілі. Цей критичний шлях є, по суті, масивом об’єктів-з’єднувачів, і масив формується шляхом перегляду кожного наступного батьківського з’єднання, доки не буде знайдено з’єднувач, що містить машини із зовнішнім доступом. На малюнку 3.9 проілюстровано цей процес. Цей критичний шлях використовується під час методології генерації атак, коли потрібно вибрати ефективний маршрут.

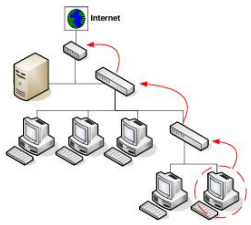

Рисунок 3.9 - Розробка критичного шляху

Генерація кроків атаки починається із зовнішнього (хакерського) розташування та просувається через мережу, зрештою досягаючи кінцевої цілі та завдання. Таким чином, цей процес працює в тій самій послідовності, в якій хакер виконав би атаку. На рисунку 3.10 показано логіку, як потік процесу, що використовується для генерації як зовнішніх, так і внутрішніх кроків атаки.

Генерація атаки розпочнеться з визначення початкової цілі, яка має зовнішній доступ. Буде вибрано відповідний шаблон інструкцій для етапу, а крок атаки буде згенеровано з використанням однієї з можливих категорій дій для конкретного етапу. Клас атаки надає спеціальний метод для генерації окремого кроку атаки з використанням певних аргументів. Ці аргументи включають вихідну та цільову машини для кроку, етап кроку, категорії дій та номер кроку. Цей метод визначатиме, який конкретний набір дій (у цьому випадку логічні експлойти) доступний з урахуванням атрибутів цільової машини та дозволів брандмауера на з'єднувачах, розташованих уздовж маршруту від джерела до цільової машини. Одна з дій буде вибрана випадковим чином для використання на кроці

атаки. Якщо дії недоступні, буде вибрано іншу категорію дій (для того ж етапу). Якщо для жодної з категорій дій етапу дії недоступні , тоді буде вибрано іншу цільову машину.

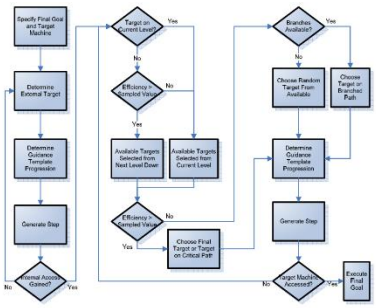

Рисунок 3.10 - Логіка генерації кроків атаки.

Після генерації кроку атаки, на якому дія успішно отримує доступ до зовнішньої машини, зовнішня частина атаки вважається завершеною. Решта методології зосереджена на внутрішніх кроках атаки (де вихідна машина є машиною в мережі). Для кожного згенерованого внутрішнього кроку атаки логіка спочатку перевіряє, чи знаходиться кінцева ціль на тому ж рівні, що й поточна вихідна машина. Якщо ціль знаходиться на тому ж рівні, то крок атаки залишатиметься на цьому рівні мережі. В іншому випадку значення ефективності використовується для визначення, чи буде на кроці атаки обрано цільову машину на наступному (глибшому) рівні мережі. Також значення ефективності буде знову використовуватися для визначення того, чи слід дотримуватися критичного шляху, чи крок повинен розгалужуватися, вибираючи з цільових машин на іншому роз'ємі.

атаки (так само, як визначено під час генерації зовнішнього кроку). Залежно від значення параметра прихованості, вибраний етап шаблону наведення може представляти цільовий етап (етап 4 або етап 9 у шаблоні наведення за замовчуванням), який генеруватиме крок атаки, що виконує проміжну дію цілі. Також значення параметра навички визначатиме, чи буде вибрано логічний експлойт як дію кроку атаки, чи буде обрано випадкову дію.

Внутрішні кроки атаки будуть генеруватися таким самим чином, доки кінцева цільова машина не стане ціллю кроку атаки. Наступний крок атаки після

цього кроку представлятиме останній крок атаки та включатиме відповідну цільову дію.

3.2 Обробка подій, моніторинг активності та аналіз інцидентів безпеки

Пакет моделювання, що надається в симуляторі, відповідає за керування та виконання набору подій, що складають прогін моделювання. У цьому підрозділі представлено більше деталей про класи, що входять до пакету моделювання, та обговорено методологію, яка використовується для моделювання сценарію атаки.

Клас сутності розроблено як абстрактний клас для представлення будь-яких транзакцій, що переміщуються мережею. У цьому застосунку клас трафіку успадковується від класу сутності та представляє єдиний тип моделюваної сутності. Клас вузла використовується як загальне представлення мережевих розташувань, і цей клас є надкласом класу мережевого обладнання. Клас вузла включає методи, які запускаються, коли сутності входять у вузол або залишають його. Клас сутності надає атрибути, що вказують, на якому вузлі знаходиться сутність на даний момент , на якому вузлі сутність була раніше (якщо така є) та на який вузол сутність переміститься далі (якщо така є). Ці атрибути необхідні при порівнянні з дозволами брандмауера, що використовуються об'єктами з'єднувача. Моделювання трафіку як сутностей, що переміщуються між вузлами в мережі, забезпечує більш реалістичний підхід до представлення функціональності мережі та дозволяє децентралізувати мережеві рішення на вузлах мережі, де приймаються реальні мережеві рішення.

Завдання реалізації сценарію атаки (за допомогою моделювання) вимагає кількох взаємодій між класами як у пакеті моделювання, так і в пакеті атаки. На рисунку 3.11 показано взаємодії між класами моделювання, порядку подій, вузлів та сутностей пакета моделювання та класами сценарію атаки та трафіку пакета атаки. Кожен клас відображається в блоці разом із ключовими методами, що

використовуються класом, а тип взаємодії між двома класами відображається у вигляді спрямованої стрілки.

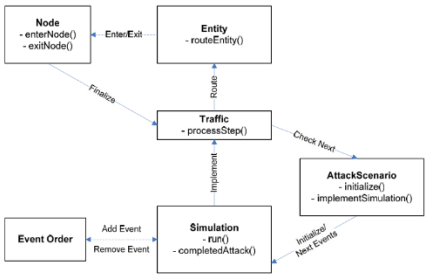

Рисунок 3.11 - Взаємодія класів пакета моделювання

Клас моделювання використовується для запуску сценарію (об'єкт моделювання створюється як атрибут сценарію атаки). Цей клас містить метод запуску, який керує часом моделювання та обробляє події, що складають сценарій атаки. Для кожної події викликається метод у класі сутності для обробки події. Однак цей метод спочатку перевантажується в класі трафіку для отримання деталей трафіку. Під час обробки трафіку запускаються відповідні методи входу та виходу задіяних вузлів. Цей процес включає маршрутизацію трафіку як через вузли машин, так і через вузли роз'ємів. Також, якщо на будь-якій з машин або роз'ємів, що входять до складу, розташовані датчики, можуть бути згенеровані об'єкти сповіщень, пов'язані з трафіком. Протягом усього цього процесу як об'єкти кроку атаки, так і об'єкти шуму перетворюються назад на об'єкти трафіку, щоб гарантувати, що ці різні типи мережевого трафіку не відрізняються вузлами, через які вони проходять. Цей метод гарантує відсутність упередженості в обробці об'єктів шуму або об'єктів кроку атаки, оскільки такі типи трафіку не можна було б безпосередньо розрізнити в реальній мережі.

Симулятор використовує форму моделювання дискретних подій для виконання сценаріїв атаки. Масив об'єктів трафіку, включаючи кроки атаки та шум, використовується як список подій для моделювання. Цей масив об'єктів трафіку є частиною класу порядку подій, який містить методи для додавання та видалення подій з масиву. Під час ініціалізації моделювання створюється новий

об'єкт порядку подій, який містить порожній масив об'єктів трафіку. Сповіщення про шум (за потреби) та перший крок атаки з кожної атаки у сценарії розміщуються в масиві порядку подій у порядку зростання часу початку. На рисунку 3.12 представлено логіку моделювання, яка використовується для керування моделюванням дискретних подій сценарію атаки.

Моделювання виконується шляхом послідовної обробки кожного об'єкта трафіку (події), що міститься в масиві порядку подій. Час моделювання спочатку починається з 0, а тактовий годинник моделювання завжди переходить до часу початку наступної події в таблиці подій. Під час обробки події можуть статися дві речі: по-перше, об'єкт трафіку події може призвести до генерації сповіщень датчиками; по-друге, подія може призвести до додавання іншої події до таблиці подій. Наприклад, якщо наступна подія в таблиці подій є кроком атаки, цей крок може призвести до генерації сповіщень датчиками. Крім того, завершення цього кроку атаки дозволяє додати наступний крок атаки до масиву подій. Події обробляються в масиві подій, доки не залишиться жодної події, що вказує на завершення моделювання сценарію атаки.

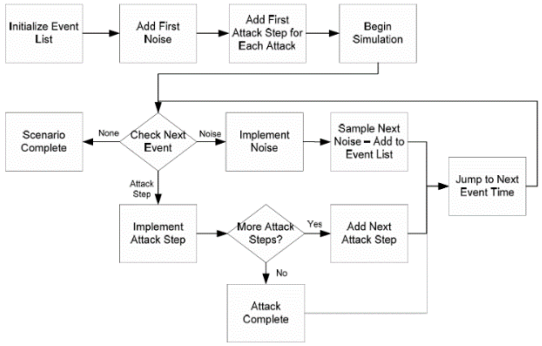

Рисунок 3.12 - Дискретне моделювання подій сценарію атаки.

Під час запуску сценарію атаки користувачеві відображається інтерфейс, що відображає події, пов'язані з атакою, що відбулися під час симуляції. Цей інтерфейс показано на рисунку 3.13. Подібно до автоматичної генерації атаки, користувач може вказати, чи використовувати загальний набір випадкових чисел, чи певне початкове значення випадкових чисел. Випадкові числа

використовуються в симуляції подій під час генерації шумових об'єктів та визначення конкретного часу початку подій, доданих до списку подій (в основному для трафіку, який використовує експоненціально розподілений час початку).

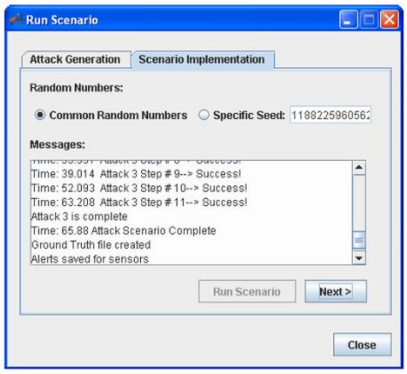

Рисунок 3.13 - Інтерфейс моделювання подій

Результати сценарію

Після завершення реалізації сценарію атаки надаються два основні вихідні дані. Ці вихідні дані включають фактичні дані сценарію та сповіщення датчиків. Після генерації цих вихідних даних користувачеві надається інтерфейс, який дозволяє переглядати файли, пов'язані з вихідними даними. На рисунку 3.14 показано цей інтерфейс, включаючи список одного вихідного файлу, що представляє фактичні дані, та чотирьох вихідних файлів, що представляють сповіщення датчиків.

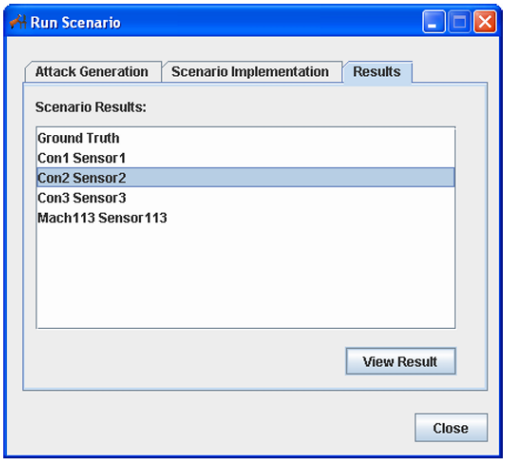

Рисунок 3.14 - Інтерфейс результатів сценарію

Вихідні дані наземних даних відображають, яка саме шкідлива активність на основі атаки фактично відбувається під час реалізації сценарію. Ця інформація в основному відповідає тому, які кроки атаки генеруються та обробляються моделлю. Наявність цієї інформації дозволяє визначити як розміщення датчиків, так і ефективність будь-якого інструменту об'єднання інформації, який використовується для оцінки.

Приклад файлу вихідних даних наземних даних показано на рисунку 3.15. У цьому файлі кроки атаки представлені в порядку їх виникнення. Перелічені атрибути включають атаку, до якої належав крок, номер кроку в атаці, категорії дій, назву дії (або повідомлення), шлях кроку (із зазначенням IP-адреси джерела та призначення) та вказівку на те, чи був крок успішним чи невдалим.

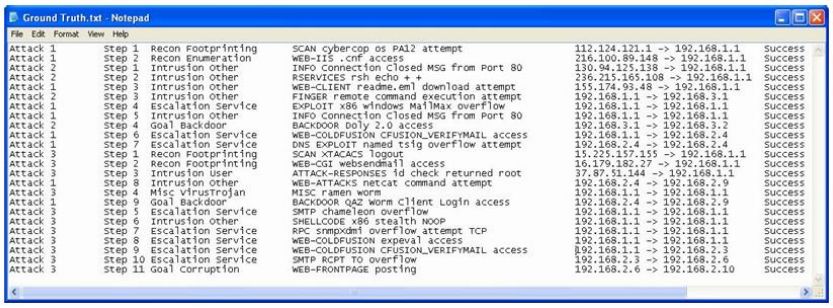

Рисунок 3.15 - Приклад файлу наземних даних

Вихідні дані сповіщень датчика представляють усі сповіщення, згенеровані датчиками під час реалізації сценарію. Ці вихідні дані включають сповіщення, згенеровані як в результаті атаки, так і в результаті шумової активності. Окремі об'єкти сповіщень використовуються для представлення кожного конкретного сповіщення, згенерованого під час запуску симуляції. Ці об'єкти сповіщень створюються датчиком мережевого пристрою, коли об'єкт (сутність) трафіку, що входить до вузла цього пристрою, має сигнатуру дії, яка відповідає сигнатурі сповіщення в базі даних сигнатур сповіщень датчика. Об'єкт сповіщення додається до масиву об'єктів сповіщень, що зберігаються як атрибут датчика. Доступ до цього масиву здійснюється в кінці симуляції під час створення вихідних файлів. Крім того, класи в пакеті керування датчиками, якщо цей пакет використ-

товується, матимуть доступ до цього масиву протягом усієї симуляції.

Вихідні файли для сповіщень датчиків складаються з одного файлу для кожного датчика, що використовується в мережі. Кожен файл містить список сповіщень у форматі, що використовується відповідним типом датчика. На рисунку 3.16 показано набір сповіщень, згенерованих мережевим датчиком виявлення вторгнень Snort. Повідомлення сповіщень форматуються так само, як і справжні повідомлення сповіщень Snort. Це повідомлення містить час виявлення, використаний порт, повідомлення про дію/сповіщення, класифікацію сповіщення, ранжований пріоритет сповіщення, використаний протокол, а також IP-адреса джерела та призначення. Класифікація та пріоритет використовуються для визначення ступеня серйозності конкретного сповіщення.

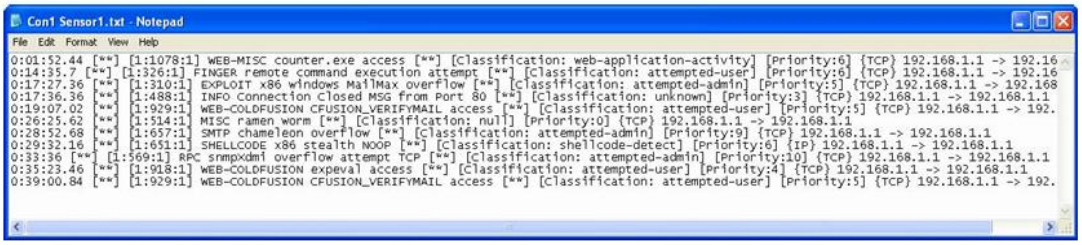

Рисунок 3.16 - Приклад файлу сповіщень датчика

Різні файли сповіщень датчиків, що генеруються, призначені для представлення сповіщень фактичної системи виявлення вторгнень. Таким чином, генеровані сповіщення не дають жодної інформації про те, чи було сповіщення пов'язане з атакою, чи це просто звичайний мережевий шум. Мета цих файлів сповіщень полягає в тому, щоб забезпечити інструменти об'єднання інформації вхідними даними сповіщень, які відображають реальні мережеві сповіщення. Ці інструменти потім можна оцінити за допомогою змодельованих сповіщень, замість того, щоб вимагати отримання фактичних даних сповіщень.

Експорт сценаріїв атак

Ще одним потенційним результатом, який може надати симулятор, є відформатований XML-документ, що описує атаки, включені до сценарію атаки.

У меню налаштувань програми передбачено опцію виведення такого XML-документа після процесу генерації атаки. Цей документ містить відповідну інформацію про всі атаки у сценарії, а також повний набір кроків атаки, використаних у атаці. На рисунку 3.17 показано XML-схему, яка використовується для цього XML-документа з інформацією про атаку.

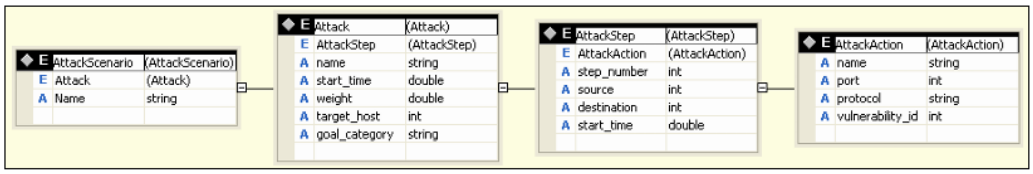

Рисунок 3.17 - XML-схема сценарію атаки.

Процес створення цього документа подібний до процесу, що використовується для створення XML-документа віртуальної місцевості. Пов'язаний об'єкт сценарію атаки, об'єкти атаки та об'єкти кроків атаки використовуються для створення елементів та атрибутів, що входять до складу XML-документа. Крім того, цей XML-документ можна імпортувати назад у мережеву модель (для тієї ж мережі), щоб створити сценарій атаки з тими ж атаками та кроками атаки. Цей процес також надає іншим програмам можливість налаштувати набір атак, які будуть згенеровані в певній мережі, яка або змодельована в симуляторі, або збережена в документі віртуальної місцевості.

3.3 Оцінка ефективності програмного забезпечення, результати експериментів та перспективи розвитку системи

Заключним етапом розробки програми для моделювання кібератаки є оцінка того, наскільки добре програма виконує задумані функції. У цьому розділі буде надано формальну перевірку та валідацію мережевих та атакуючих компонентів, що використовуються в моделі. Також обговорюються можливості, обмеження та застосування моделі в інших суміжних галузях досліджень.

Як мережева модель, так і процес моделювання атаки потребують належної оцінки, щоб забезпечити їхнє функціонування відповідно до представлених методологій. У підрозділі обговорюються методи та експерименти, що використовуються для перевірки того, що функції моделювання мережі програми правильно визначають детальну віртуальну мережу. Аналогічно, обговорюються методи та експерименти, що використовуються для перевірки того, що процес генерації атаки відповідає шаблону інструкцій та методології автоматичної атаки, а також що моделювання подій фактично спрямовує та обробляє події сутностей належним чином через змодельовану мережу.

Обговорюється процес перевірки як атак, так і сповіщень, згенерованих за допомогою моделі. Цінність результату, отриманого цією моделлю, значною мірою залежить від атак, що представляють реальні кібератаки в мережі, та сповіщень IDS, що представляють реальні сповіщення датчиків, що виникають в результаті як атак, так і мережевого шуму.

Після обговорення питань перевірки та валідації моделі, представлено загальні можливості моделі. Ці можливості включають те, для чого можна використовувати модель, а також які дані вона може приймати як вхідні. Незважаючи на численні можливості, модель також має кілька обмежень, які можна розглядати як перешкоду для забезпечення більш детального та складного моделювання, так і як вказівку на те, які функції слід додати для покращення моделі.

Зрештою, сама по собі модель не забезпечує значної цінності для вирішення проблем у кіберсфері. Інтеграція цієї програми з іншими програмами, такими як інструменти об'єднання інформації, необхідна для досягнення помітних успіхів як у моделюванні, так і у виявленні кібератак. Зазначено деякі інші програми, з якими використовується або призначений для використання цей симулятор кібератак.

Перевірка симуляції атаки

Перевірка того, що атаки створені та змодельовані належним чином, необхідна для того, щоб результати, отримані в результаті симуляції, відображали те, що задумано на основі вхідних даних, наданих користувачем. Ця перевірка

виконується за допомогою шести різних експериментів. Перший експеримент розроблений для того, щоб забезпечити правильну фільтрацію доступного набору експлойтів на основі атрибутів машин та роз'ємів, що беруть участь у виконанні експлойта. Другий експеримент відстежує етапи, які проходить автоматично згенерована атака, щоб перевірити правильність дотримання шаблону інструкцій. Підрозділ містить обговорення експерименту, який перевіряє використання шаблону інструкцій. Третій експеримент оцінює вплив трьох параметрів автоматичної атаки, щоб переконатися, що ці параметри належним чином використовуються процесом генерації атаки. Четвертий експеримент перевіряє, чи об'єкти належним чином маршрутизуються між цільовими вузлами мережевої моделі, і цей експеримент описано. Шостий експеримент перевіряє, чи шум генерується належним чином. Заключний експеримент порівнює набір сповіщень, згенерованих датчиками, з мережевим трафіком, обробленим під час моделювання, щоб перевірити, чи для конкретних датчиків генерується правильний набір сповіщень.

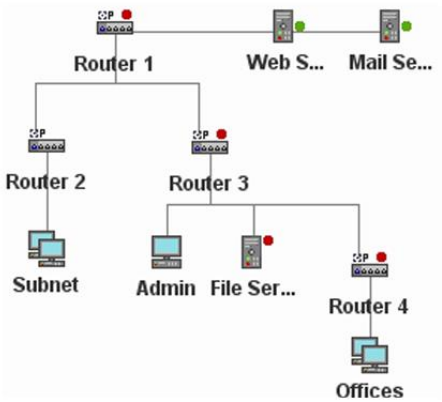

Рисунок 3.18 - Приклад мережі для перевірки атак

У всіх цих різних експериментах використовується одна й та сама мережева модель. Ця мережева модель розроблена на основі моделі з рисунка 3.18. Ця модель включає додатковий мережевий рівень, а також IDS датчики, розміщені на маршрутизаторі 1, маршрутизаторі 3, маршрутизаторі 4 та файловому сервері. Для зручності уточнення цілей кроків атаки, що використовувалися в

експериментах, IP-адреса або діапазон IP-адрес кожної машини в цій мережевій моделі наведено в таблиці 3.2.

Фільтрація експлойтів

Під час налаштування кроку ручної атаки фільтрація нелогічних експлойтів є дуже корисною функцією. Результати цього процесу повинні включати лише експлойти, можливі на основі атрибутів цільової машини та шляху трафіку кроку атаки. Перевірка того, що надано правильні експлойти, включає перевірку деталей кожного експлойта на відповідні атрибути машини та роз'єму. Для перевірки процесу вибору експлойта налаштовується експеримент з використанням мережевої моделі. Експеримент полягає в налаштуванні кроку ручної атаки з веб-сервера (з IP-адресою 192.168.1.1) на файловий сервер (з IP-адресою 192.168.3.2) та спостереженні як за загальним набором перелічених експлойтів, так і за логічним набором експлойтів, перелічених як можливі дії для атаки. Деталі кроку атаки відображені на рисунку 3.19.

Таблиця 3.2 -

IP-адреси, що використовуються в прикладі мережі

Машина / Вузол (Machine)	**IP-адреса або діапазон (IP Address or Range)**
Веб-сервер (Web Server)	192.168.1.1
Поштовий сервер (Mail Server)	192.168.1.2
Підмережа (Subnet)	192.168.2.1 – 192.168.2.100
Адміністратор (Admin)	192.168.3.1
Файловий сервер (File Server)	192.168.3.2
Офіси (Offices)	192.168.4.1 – 192.168.4.20

Для категорії дій «Розвідка/Сканування» представлено досить великий список потенційних експлойтів. Однак, під час вибору опції фільтрації нелогічних експлойтів представлено лише вісім конкретних експлойтів. Ці вісім експлойтів відповідають деталям цільової машини. Наприклад, потрібна операційна система – Linux, потрібний тип машини – сервер, а ідентифікатори

вразливостей експлойтів пов'язані зі службою ICMP. Ці три атрибути відповідають атрибутам об'єкта машини файлового сервера. Крім того, порт, пов'язаний зі службою ICMP, дозволений дозволами брандмауера вздовж шляху від веб-сервера до файлового сервера. Таким чином, фільтрація експлойтів виконується належним чином.

Прогресія етапу

Процес автоматичного створення кроків атаки значною мірою залежить від правильного використання шаблону інструкцій. Шаблон інструкцій контролює, які етапи повинні передувати яким іншим етапам, а атаки, згенеровані за допомогою симулятора, повинні демонструвати цей пріоритет етапів. Було проведено експеримент для перевірки реалізації шаблону інструкцій за допомогою мережевої моделі.

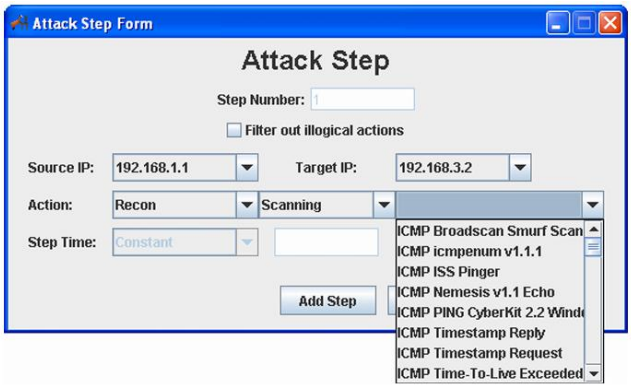

Рисунок 3.19 - Деталі кроку атаки

Експеримент з просуванням по етапах використовує сценарій атаки з дев'ятьма окремими автоматичними атаками, які мають різні значення ефективності, значення скритності, цільові машини та категорії цілей. Характеристики кожної атаки наведено в таблиці 3.3. Ці характеристики варіюються, щоб врахувати різну складність та ситуації, з якими може зіткнутися процес автоматичної генерації атак. Атаки генеруються та обробляються під час запуску моделювання, а отримані наземні дані спостерігаються для перевірки відповідності просування по етапах.

Базова істина показує, що категорії дій та відповідні етапи, що використовуються в кожній атаці, відповідають правильній послідовності шаблону інструкцій. Наприклад, Атака 2 складається з дванадцяти дій, які відбуваються в послідовності, що відповідає пріоритету етапів, визначеному в класі шаблону інструкцій, і ці дії відображаються в таблиці.

Таблиця 3.3

Деталі атаки для експерименту з прогресуванням етапів

Назва (Name)	**Ефективність (Efficiency)**	**Скритність (Stealth)**	**Навичка (Skill)**	**Ціль (Target)**	**Категорія цілі (Goal Category)**
Attack 1	0.2	0.9	1.0	192.168.2.59	Шпигунство (Espionage)
Attack 2	0.5	0.9	1.0	192.168.4.16	Відмова в обслуговуванні (DOS)
Attack 3	0.9	0.9	1.0	192.168.3.1	Корупція/Псування даних (Corruption)
Attack 4	0.2	0.5	1.0	192.168.2.24	Розкрадання (Pilfering)
Attack 5	0.5	0.5	1.0	192.168.1.1	Бекдор (Backdoor)
Attack 6	0.9	0.5	1.0	192.168.3.2	Бекдор (Backdoor)
Attack 7	0.2	0.2	1.0	192.168.7.16	Шпигунство (Espionage)
Attack 8	0.5	0.2	1.0	192.168.2.1	Корупція/Псування даних (Corruption)
Attack 9	0.9	0.2	1.0	192.168.4.11	Відмова в обслуговуванні (DOS)

Ця атака починається з дії слідування (етап 0), а потім вторгнення/іншої дії, спрямованої на веб-сервер із зовнішнього розташування. Після отримання доступу до веб-сервера атака продовжується різними діями/діями, пов'язаними з вірусами-троянами та розвідкою/переліком, які виконують додаткову шкідливу активність проти веб-сервера та встановлюють черв'яків. Ці дії відповідають етапу 5 прогресії шаблону інструкцій. Далі, проти машини в групі підмережі виконується дія вторгнення/корінного доступу (етап 6), щоб отримати доступ до машини підмережі. З цього моменту виконуються дія розвідки/переліку (етап 5) та дія ескалації/обслуговування (етап 7), спрямовані на машину адміністратора та отримання доступу до цієї машини. Дія ескалації/обслуговування виконується

знову (внутрішньо), щоб отримати додатковий доступ, а потім виконується ще одна дія ескалації/обслуговування проти машини в групі офісів. Потім отримується доступ до кінцевої цільової машини атаки (192.168.4.16) за допомогою ще однієї дії ескалації/обслуговування. Зрештою, кінцева мета атаки досягається за допомогою дії goal/dos (етап 9).

Етапи, що використовуються діями, що виконуються під час атаки, відстежуються для визначення послідовності, в якій відбуваються ці етапи. Ця послідовність така: етап 0, етап 1 та етап 5 (три рази), етап 8, етап 5, етап 7 (чотири рази) і, нарешті, етап 9. Ця послідовність атак повністю відповідає правилам, визначеним у класі шаблонів інструкцій див. табл. 3.4.

Таблиця 3.4

Кроки атак, згенеровані для атаки 2

Крок (Step)	**Категорія дії (Action Category)**	**Дія (Action)**	**Шлях (Path)**
Крок 1	Розвідка / Збір інформації	RPC portmap sadmind request UDP	209.153.125.235 -> 192.168.1.1
Крок 2	Вторгнення / Інше	SHELLCODE x86 stealth NOOP	212.97.19.144 -> 192.168.1.1
Крок 3	Різне / Вірус / Троян	MISC ramen worm	192.168.1.1 -> 192.168.1.1
Крок 4	Розвідка / Перерахування	FINGER account enumeration attempt	192.168.1.1 -> 192.168.1.1
Крок 5	Різне / Вірус / Троян	MISC ramen worm	192.168.1.1 -> 192.168.1.1
Крок 6	Вторгнення / Root	MS-SQL/SMB xp_showcolv possible buffer over	192.168.1.1 -> 192.168.2.22
Крок 7	Розвідка / Перерахування	WEB-PHP read_body.php access attempt	192.168.2.22 -> 192.168.3.1
Крок 8	Підвищення привілеїв / Служба	WEB-MISC changepw.exe access	192.168.2.22 -> 192.168.3.1
Крок 9	Підвищення привілеїв / Служба	SMTP VRFY overflow attempt	192.168.3.1 -> 192.168.3.1
Крок 10	Підвищення привілеїв / Служба	WEB-COLDFUSION CFUSION_VERIFYMAIL access	192.168.3.1 -> 192.168.4.8
Крок 11	Ціль / DoS	DDOS TFN client command BE	192.168.4.16 -> 192.168.4.16

Параметри атаки. Три параметри автоматичної генерації атак призначені для забезпечення значного рівня контролю над генерацією атак шляхом простої зміни значень параметрів від 0 до 1. Окремий експеримент проводиться для перевірки того, що кожен параметр забезпечує цей рівень контролю.

Параметр ефективності впливає на прямоту та складність атак. Очікується, що високоефективні атаки складатимуться лише з кількох кроків, які всі йдуть критичним шляхом до цілі, тоді як дуже неефективні атаки матимуть багато кроків, що проходять через мережу. Експеримент, проведений для перевірки цього результату, встановлюється з використанням мережевої моделі, показаної на рисунку 3.20, і складається з двадцяти атак, що використовують ту саму кінцеву ціль та категорію завдань. Для цих атак параметр ефективності варіюється від 0,05 до 1,0, тоді як інші параметри залишаються на рівні 1,0.

Оскільки значення ефективності є єдиною різницею між атаками, кількість кроків та складність атак можуть бути використані для визначення ефективності параметра. Використовується десять повторень цього сценарію (з різним випадковим початковим значенням для кожного повторення), щоб забезпечити достатню кількість даних, пов'язаних з кроками, для кожного значення параметра. На рисунку 3.20 показано кількість кроків, згенерованих для кожного з десяти повторень для всіх використаних значень ефективності. Лінія тренду відображається для позначення кореляції параметра ефективності з кількістю кроків.

Визначено, що нижчі значення ефективності значно призводять до збільшення кількості кроків атаки. Тенденція на рисунку 3.20 показує, що експеримент дає кореляцію 80,5% між ефективністю та кількістю згенерованих кроків. Спостереження за окремими прогресіями атаки, розробленими для кожного значення ефективності, також вказує на те, що атаки з низьким значенням ефективності мають складніший шлях атаки, ніж атаки з високим значенням ефективності. Наприклад, дев'ять з десяти повторень атаки, пов'язаної з ефективністю 0,9, просуваються вздовж шляху критичного з'єднувача до кінцевої цілі (у групі Office). Однак шість повторень атаки, пов'язаної з ефективністю 0,3, розгалужуються до групи Subnet, перш ніж перейти до машин, підключених до

маршрутизатора 3. Кількість та складність кроків атаки, створених для цього експерименту, вказують на те, що параметр ефективності має відповідний вплив на процес автоматичної генерації атаки.

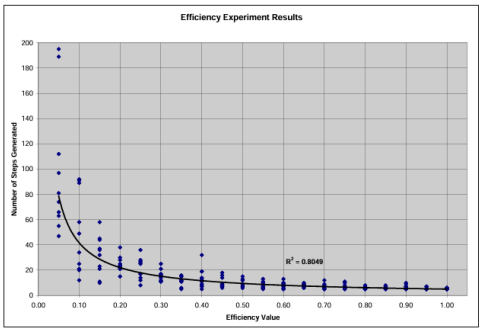

Рисунок 3.20 - Зв'язок між ефективністю та кількістю кроків.

Параметр скритності впливає на те, наскільки добре уникаються проміжні кроки цілі під час прогресу атаки, і ці кроки з більшою ймовірністю будуть виконані при використанні низького значення скритності, ніж при використанні високого значення скритності. Експеримент, який використовувався для перевірки правильної реалізації параметра скритності, також використовує мережеву модель на рисунку.

Подібно до експерименту з ефективності, сценарій цього експерименту складається з двадцяти атак, де змінюється лише параметр прихованості (від 0,05 до 1,0). Також виконується десять повторень для отримання значної кількості даних. Основною метрикою в згенерованих атаках є кількість проміжних кроків цілі на атаку. Рисунок відображає цю метрику для різних значень параметрів реплікації та прихованості. Рисунок 3.21 ілюструє, що нижчі значення прихованості загалом призводять до більшої кількості проміжних кроків цілі. Однак, зв'язок між параметром прихованості та Цей показник не дуже добре відповідає жодному конкретному розподілу. Проте, значущість параметра можна перевірити , порівнявши результати при використанні нижчих значень параметрів з результатами при використанні вищих значень параметрів. Для цього порівняння виконується парний t-тест, який порівнює кількість кроків до мети, зроблених з вико-

ристанням помірно низьких значень скритності (від 0,1 до 0,3), з кількістю кроків до мети, зроблених з використанням помірно високих значень скритності (від 0,7 до 0,9). Тест з 99,9% впевненістю показує, що результати суттєво відрізняються.

-

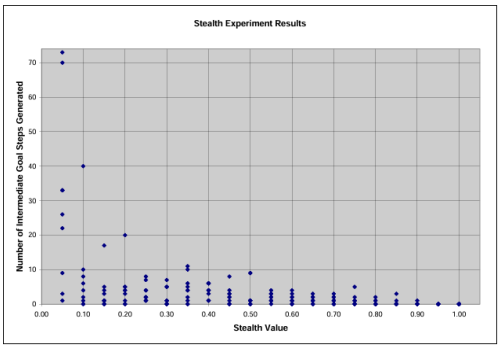

Рисунок 3.21 - Зв'язок між прихованістю та кількістю проміжних кроків цілі.

Параметр навичок впливає на ймовірність успіху кожної атаки. Висококваліфіковані атаки зазвичай будуть успішними, тоді як некваліфіковані атаки, ймовірно, зазнають невдачі в певний момент (кроку). Ця ймовірність перевіряється за допомогою експерименту з використанням мережевої моделі на рисунку 3.19 та сценарію атаки з двадцяти атак з різною ефективністю та цілями. Експеримент включає виконання десяти повторень сценарію атаки для кожного з дев'яти можливих значень параметра навичок у діапазоні від 0,1 до 0,9. Основною метрикою цього експерименту є кількість успішних атак (з двадцяти можливих атак) у сценарії атаки. На рисунку 3.22 показано графік цієї метрики, що вказує на кількість завершених атак для кожного з дев'яноста запуску моделювання. Підібрана крива ілюструє досить позитивну лінійну залежність між параметром навичок та успіхом атаки. Однак цей параметр не призначений для строгого представлення ймовірності успіху атаки. Значення навичок лише приблизно вказує на те, наскільки успішними є атаки відносно інших значень навичок (більш успішними чи менш успішними).

Для параметра навички, як і для інших параметрів, результати будуть різними залежно від використовуваної мережі. У прикладах, розглянутих у цьому

розділі, для кожного експерименту використовується однакова конфігурація мережі. Однак різні змодельовані мережі можуть призвести до різних зв'язків між параметрами.

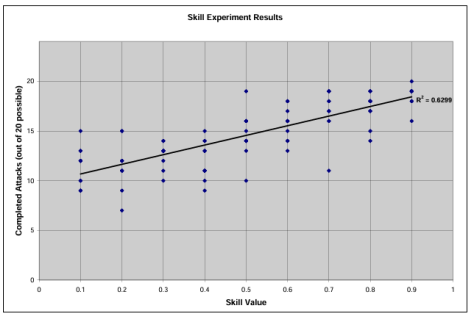

Рисунок 3.22 - Зв'язок між навичкою та завершеними атаками.

Наприклад, значення навички 0,6 у простій та досить незахищеній мережі може призвести до успіху більшості атак, тоді як значення навички 0,6 у складній та захищеній мережі може призвести до успіху лише невеликого відсотка атак. Аналогічно, ефективність 0,3 у великій та складній моделі мережі зазвичай призведе до набагато більшої кількості кроків атаки, ніж така ж ефективність, що використовується в моделі невеликої мережі. Автоматичні параметри атаки призначені просто для керівництва процесом генерації, а не для надання конкретних результатів, і різні значення цих параметрів ефективно показують, що встановлюють різні атаки, якщо розглядати загальну мережу.

Генерація сповіщень

Сповіщення IDS генеруються під час запуску моделювання, коли об'єкти трафіку переміщуються через вузли мережі, з якими пов'язані датчики IDS. Набір сповіщень, згенерованих для кожного датчика, стає основними вихідними даними в кінці запуску моделювання. Для того, щоб генерувати точні сповіщення, модель повинна мати можливість ідентифікувати сигнатуру дії, пов'язану з трафіком, і зіставити сигнатуру з базою даних сповіщень датчика, щоб визначити, чи потрібно генерувати сповіщення. Для перевірки того, що ці завдання виконуються належним чином, використовується експеримент.

Таблиця 3.5

Сповіщення про кроки атаки

Крок атаки (Attack Step)	**Дія (Action)**	**Сповіщення сенсорів (Sensor Alerts)**
1	SCAN cybercop os PA12 attempt	Сенсор роутера 1, Сенсор веб-сервера
2	WEB-IIS .cnf access	Сенсор роутера 1, Сенсор веб-сервера
3	WEB-CLIENT readme.eml download attempt	Сенсор роутера 1, Сенсор веб-сервера
4	BACKDOOR Infector 1.6 Server to Client	Сенсор веб-сервера
5	INFO Connection Closed MSG from Port 80	Сенсор веб-сервера
6	WEB-MISC changepw.exe access	Сенсор веб-сервера, Сенсор роутера 1, Сенсор роутера 3
7	WEB-PHP shoutbox.php directory traversal attempt	Немає сповіщень
8	DNS EXPLOIT named overflow attempt	Сенсор роутера 3, Сенсор роутера 4
9	POLICY SMTP relaying denied	Немає сповіщень

Експеримент для цієї процедури перевірки включає той самий сценарій атаки, що й для експерименту з маршрутизацією. З мережевих пристроїв, що беруть участь у сценарії атаки, маршрутизатор 1, маршрутизатор 3, маршрутизатор 4 та веб-сервер містять датчики IDS. Для кожного кроку атаки перевіряється шлях, щоб визначити, чи знаходиться будь-який з цих чотирьох пристроїв вздовж шляху трафіку. Для кожного пристрою вздовж шляху перевіряється вихідний сигнал датчика IDS (після запуску моделювання), щоб переконатися, що сформовано сповіщення та що воно відповідає дії кроку атаки. У таблиці 3.5 показано дію, пов'язану з кожним кроком атаки, та пристрої, які генерують сповіщення для цієї дії.

Кроки атаки сім і дев'ять не створюють жодних пов'язаних сповіщень, оскільки вздовж шляху цих кроків немає датчиків. Для всіх інших кроків відповідні датчики генерують сповіщення, які вказують на виконання правильної дії. Крім того, різниця між NIDS та HIDS перевіряється шляхом повторного запуску сценарію з нижчим значенням навички. З цією модифікацією один із кроків атаки, спрямованих на веб-сервер, не виконується. Датчик, пов'язаний з маршру-

тизатором 1, все ще генерує сповіщення, яке вказує на те, що було зроблено спробу дії, але датчик, пов'язаний з веб-сервером, не генерує сповіщення, оскільки дія не була успішно виконана.

Перевірка атак та сповіщень

Для забезпечення точного виведення сповіщень за допомогою симуляційної моделі, генерація змодельованих атак та сповіщень датчиків повинна відповідно достовірно відображати реальні мережеві атаки та сповіщення датчиків. В ідеалі, процес валідації включає порівняння результатів симулятора з інформацією, доступною в реальній мережевій конфігурації, щоб перевірити відповідність між змодельованими результатами та реальними результатами мережі. Однак така реальна мережева конфігурація повинна забезпечувати апріорні знання про атаки, що відбуваються, разом з усім набором сповіщень IDS. Дуже мало існуючих мереж надають таку можливість, а мережі, які надають цю можливість, зазвичай зберігають результати у власності. З цих причин необхідно застосувати інший підхід, який передбачає логічне порівняння згенерованих атак та сповіщень з тим, що очікується в реальній мережі. У решті цього розділу буде розглянуто, як конкретні аспекти симуляційної моделі перевіряються як такі, що представляють відповідні аналоги в реальній мережі.

Забезпечення імітації зв'язку між пристроями в мережі є необхідним компонентом коректного представлення атак. Симулятор моделює цей зв'язок за допомогою сутностей трафіку, які представляють набір мережевих пакетів, пов'язаних з однією конкретною дією (оскільки дії можуть бути розподілені по кількох пакетах). Одне й те саме представлення використовується як для дій на основі атаки, так і для дій на основі шуму. Під час симуляції сутності трафіку маршрутизуються через відповідні вузли в мережевій моделі так само, як фактичний трафік маршрутизується через пристрої у фізичній мережі. Крім того, дії, що виконуються сутностями трафіку, обмежені атрибутами цільової машини (такими як служби та операційна система) та дозволами задіяних роз'ємів. Обмеження трафіку таким чином відображає, як реальні комп'ютерні мережі вразливі не просто до будь-яких дій, а до дій, що відповідають вразливостям комп'ютерів та

пристроїв маршрутизації в мережі. Таким чином, мережева модель перевірена як така, що надає репрезентативні сутності трафіку та відповідні деталі мережі, які ефективно обмежують дії, що виконуються з сутностями трафіку.

Щодо генерування атак, параметри автоматичної атаки та вхідні дані шаблону вказівок дозволяють процесу генерування атак створювати бажані типи атак. Оскільки інтеграція шаблону вказівок перевірена як така, що функціонує належним чином, ці вхідні дані можна скоригувати або оновити, щоб відобразити точнішу або детальнішу прогресію етапів атаки. Використовуючи цей підхід, експерт у предметній області може чітко визначити типи атак, можливі за допомогою симулятора, щоб гарантувати, що генеруються лише дійсні атаки. Поточні вхідні дані шаблону вказівок використовують прогресії етапів, які відповідають графовому пріоритету етапів, визначеному експертами у предметній області. Таким чином, модель є дійсною щодо класифікації та пріоритету етапів атак експертами у предметній області, а гнучкість вхідних даних шаблону вказівок симулятора дозволяє оновлювати або уточнювати етапи атаки за потреби. Однак підтримка цих вхідних даних шаблону вказівок є необхідною частиною забезпечення як дійсних, так і актуальних атак за допомогою симулятора.

У реальній мережі датчики IDS не здатні розрізняти дії, пов'язані з атакою, від дій, які є просто звичайним, нешкідливим мережевим шумом. Ця ж особливість датчиків IDS імітується в симуляційній моделі. Коли кроки атаки або шумові дії маршрутизуються між вузлами в мережі, пов'язані сутності перетворюються на сутності трафіку. Клас трафіку, що використовується в цій моделі, є надкласом як кроку атаки, так і класу шуму; таким чином, об'єкт трафіку все ще містить всю необхідну інформацію для маршрутизації сутностей. Кожен датчик IDS запускає сповіщення, коли сутність трафіку, що входить до вузла, що містить датчик, пов'язана з сигнатурою сповіщення у базі даних визначень сповіщень відповідного датчика . Протягом усього цього процесу датчики IDS не отримують жодної інформації про те, чи пов'язана сутність з атакою чи з шумом. Таким чином, об'єкти датчиків IDS коректно реагують на змодельований мережевий трафік так само, як реальний IDS реагує на реальний мережевий трафік.

Вихідні дані сповіщень моделі моделювання призначені для представлення реальних сповіщень, які були б згенеровані під час виконання сценарію атаки в реальній мережі. З цієї причини сповіщення, згенеровані змодельованими датчиками, повинні надавати ту саму інформацію, що й реальні сповіщення датчиків. Зокрема, для сповіщень Snort ця інформація складається з позначки часу, повідомлення сповіщення, протоколу, класифікації, рангу пріоритету та мережевого шляху. Вхідні дані визначень сповіщень, що використовуються змодельованими датчиками Snort, містять дані сповіщень, зібрані з використанням інформації як з веб-сайту Snort, так і з тестових мережевих даних сповіщень Skaion, які містять реальні сповіщення датчиків Snort. Таким чином, вся та сама інформація про сповіщення, що надається в реальних сповіщеннях Snort, доступна для використання змодельованими сповіщеннями. Крім того, змодельовані сповіщення Snort використовують той самий формат повідомлень та виводу, що й реальні сповіщення, що дозволяє обробляти змодельовані сповіщення так само, як і реальні. Крім того, симулятор надає додатковий компонент (пакет керування датчиками), який можна використовувати для обмеження виводу сповіщень під час запуску моделювання. Цей тип функціональності потенційно може бути використаний у реальній мережі, де пропускна здатність та інші обмеження змушують датчики мережі виводити лише частину всіх генерованих сповіщень. Усі ці міркування щодо генерації сповіщень гарантують, що симулятор генерує дійсні сповіщення датчиків.

Можливості моделі

Модель моделювання має багато можливостей, що враховують як початкові наміри, зазначені в описі задачі, так і інші питання, що стали важливими. Основні можливості моделі включають моделювання детальних мереж та моделювання сценаріїв атак і систем виявлення вторгнень у таких мережах. Компетентний користувач здатний ефективно використовувати функції моделювання, імітувати бажані сценарії атак та використовувати результати виявлення вторгнень. У цьому розділі будуть розглянуті конкретні можливості, які надає модель.

Модель моделювання дозволяє легко (і графічно) визначати як великі, так

і складні мережі. Такі мережі можуть мати багато рівнів глибини та містити підмережі з сотнями хост-машин. На рисунку 3.22 ілюструється одна з таких великих мережевих моделей, що складається з п'яти мережевих рівнів та понад 1500 машин. Функціональність моделювання мережі дозволяє кожній машині, роз'єму та підмережі мати унікальний набір атрибутів разом із датчиками IDS, які можна легко вмикати та вимикати. Крім того, залучені роз'єми можуть визначати певні дозволи для кожного унікального шляху з'єднання між мережевими пристроями. Мережа, зображена на рисунку 3.22, швидко зберігається та завантажується, а деталі мережі також легко доступні під час налаштування сценарію атаки.

Можливість імпортувати модель мережі з документа віртуальної місцевості або експортувати змодельовану мережу до документа віртуальної місцевості є ще однією корисною можливістю симуляційної моделі. Така взаємодія з документами віртуальної місцевості по суті дозволяє використовувати змодельовані мережі зовнішніми програмами для різних цілей. Аналогічно, мережі, призначені для використання симулятором, можуть бути визначені в зовнішній програмі та імпортовані. Незалежно від зовнішніх програм, використання імпорту та експорту віртуальної місцевості забезпечує альтернативний спосіб зберігання змодельованих мереж. Крім того, мережі, що зберігаються таким чином, можуть бути завантажені навіть тоді, коли до симулятора вносяться зміни, пов'язані з програмою. У таких випадках типові об'єктні файли, що використовуються для зберігання моделей мереж, можуть бути несумісними.

Вхідні дані шаблону інструкцій є частиною гнучкості, що надається моделлю моделювання. Хоча шаблон інструкцій відповідає за можливості генерації атак симулятора, його також можна перевизначити різними способами, щоб змінити вимоги до прогресії атаки. Таким чином, шаблон інструкцій можна оновити, щоб відобразити більш детальну прогресію атаки та при цьому залишатися сумісним із моделлю моделювання.

Опції моделювання атак є основною можливістю симуляційної моделі. Користувач може вказувати окремі атаки на різних рівнях деталізації. Дуже детальні атаки можна створювати, вказуючи кожну дію, що виконується протягом

атаки. Ширші атаки можна створювати, надаючи кілька параметрів атаки, що відображають прямоту, прихованість та рівень майстерності хакера, який виконує атаку. Можна навіть згенерувати цілий сценарій атаки на основі набору параметрів, які включають частоту атак (у атаках за годину), середню ефективність, стандартне відхилення ефективності, середню тривалість атаки та загальну тривалість сценарію (хвилини для виконання сценарію). Ручні та автоматизовані атаки не мають обмежень щодо розміру, тому кількість кроків, що складають атаку, по суті обмежується лише обчислювальною потужністю машини, на якій запущено програму, або терпінням користувача під час налаштування атак. Наприклад, під час налаштування автоматично згенерованої атаки в мережі, показаній на рисунку 3.22, з використанням дуже низького значення ефективності (0,005), атака з 2009 кроками атаки генерується за лічені секунди. Ця атака також моделюється в мережі лише за кілька секунд. Загальний розмір сценарію атаки (як за кількістю атак, так і за кроками) також необмежений у моделі моделювання. Використовуючи опцію автоматизованого сценарію атаки, генерується сценарій атаки зі 197 атаками, кожна з яких має від 5 до 91 кроку. Всі ці атаки також генеруються та моделюються протягом кількох секунд. Після ручного визначення атак у сценарії або запуску процесу генерації атак для атак у сценарії, весь сценарій атаки можна зберегти у вигляді XML-файлу для подальшого використання та спостереження.

Окрім моделювання конкретних атак та сценаріїв атак, імітаційна модель також здатна моделювати шумовий трафік, що виникає в мережі. Цей шум являє собою мережеву активність, яка жодним чином не є шкідливою, але все ще може спричиняти хибні спрацьовування (хибні сповіщення) при перевірці датчиками IDS. Оскільки більшість сповіщень, що генеруються реальною системою IDS, є шумом, імітаційна модель повинна мати можливість генерувати та обробляти достатню кількість шумового трафіку. Для перевірки меж процесу генерації шуму встановлено експеримент. Цей експеримент полягає у запуску сценарію атаки з прикладу автоматизованої генерації сценаріїв (зі 197 атаками) разом із 10 000 шумових дій, що генеруються на годину, з розподілом за категоріями, відображеним у таблиці 8.11. Загальний час моделювання для запуску становить 84,5

хвилини, і цей сценарій оброблено до завершення приблизно протягом чотирьох фактичних хвилин. Запуск сценарію з шумом займає значно більше часу через велику кількість згенерованих сповіщень; особливо враховуючи, що більшість шумової активності відбувається у зовнішній частині мережі, де всі три пристрої (маршрутизатор 1, HTTP-сервер та зовнішній поштовий сервер) містять датчики IDS та генерують сповіщення для кожної активності. Наприклад, лише HTTP-сервер згенерував 8055 сповіщень протягом 84,5 хвилин моделювання. Однак сповіщення все одно генеруються протягом розумного часу, враховуючи тривалість та масштаб сценарію, а також обсяг шумової активності.

Таблиця 3.6

Розподіл шумової активності за категоріями

Категорія	**Розвідка (Recon.)**	**Вторгнення (Intrusion)**	**Підвищення привілеїв (Escalation)**	**Ціль (Goal)**	**Різне (Misc.)**
%	70%	10%	10%	5%	5%

Основні можливості виведення даних моделювання включають сповіщення датчиків IDS та наземні дані про сценарій. Сповіщення датчиків вказують на те, яка активність, як шкідлива, так і заснована на шумі, виявлена під час запуску моделювання. Моделювання створює окремий файл сповіщень для кожного активного датчика, розміщеного в змодельованій мережі. Ці файли сповіщень створюються з використанням загальноприйнятого текстового формату для кожного сповіщення, і модель не встановлює конкретних обмежень на розмір файлів сповіщень. Така конфігурація дозволяє легко організовувати, досліджувати та використовувати всі сповіщення, згенеровані в сценарії, як вхідні дані для інших програм. Створені наземні дані представляють усі дії, виконані в сценарії атаки, які пов'язані зі шкідливою активністю. Надання цих даних дозволяє проводити порівняння з виходом сповіщень датчика, щоб визначити, які дії не вдалося виявити за допомогою IDS. Ці наземні дані також можна використовувати в інших програмах для порівняння з сповіщеннями датчиків з тих самих

або інших причин.

Обмеження моделі

Незважаючи на численні можливості моделі, існує кілька обмежень, які впливають на її використання та застосовність. Багато з обмежень безпосередньо не впливають на достовірність моделі, але усунення цих обмежень може дозволити моделювати детальніше мережі, атаки та сповіщення за допомогою представленого застосунку. У цьому розділі будуть розглянуті аспекти імітаційної моделі, які обмежені щодо бажаної функціональності, включаючи функції моделювання роз'ємів, інтегровані типи датчиків, специфікацію автоматичних атак та базу даних експлойтів та вразливостей.

Хоча функції моделювання мережі надають користувачеві різноманітні варіанти, представлення роз'ємів все ще досить узагальнене. Модель мережі представляє роз'єм як пристрій, який дозволяє іншим підключеним пристроям взаємодіяти.

Однак насправді існує безліч типів роз'ємів (маршрутизатори, концентратори, комутатори тощо), кожен з яких має певну функціональність. Тому модель обмежена при побудові мережевої моделі, яка представляє мережу, що складається з низки певних типів роз'ємів та функцій. Також одним з основних припущень моделі є те, що певні шкідливі дії не переміщуються більше ніж на один рівень роз'єму в мережі одночасно. Однак, завдяки використанню маршрутизаторів та інших більш функціональних типів роз'ємів, які можуть визначити відповідний шлях через мережу, така діяльність легко можлива в реальній мережі.

Поточний вхідний сигнал датчика, що використовується в моделі, також досить обмежений. Snort IDS – це єдиний тип екземпляра датчика, який можна додати до мережевого пристрою. Крім того, функціональність датчика Snort більш специфічна для NIDS, ніж для HIDS. Однак, в ідеалі, модель повинна представляти NIDS та HIDS такими, якими вони насправді функціонують. Наразі єдина функціональна відмінність між класами датчиків полягає в тому, що HIDS не виявлятимуть невдалі дії, тоді як NIDS – так. Однак справжні HIDS також надають більш конкретну інформацію про виконану дію та її вплив на хост, яка

не відображається в симуляційній моделі.

Простота наданих параметрів автоматичної атаки ілюструє ще один спосіб обмеження моделі моделювання. Генерація атак з використанням лише параметрів (вичерпно) враховує підмножину потенційних типів атак, можливих у мережі. Крім того, деякі параметри призводять до неконтрольованого ступеня мінливості генерованих атак. Наявність мінливості в процесі генерації атак необхідна для зменшення упередженості користувачів та врахування альтернативних прогресій атак, але відсутність контролю, що проявляється в цій мінливості, ефективно зменшує специфічний вплив кожного параметра.

Зрештою, база даних експлойтів та зіставлень вразливостей, що використовується моделлю, навряд чи є вичерпною порівняно з базами даних, що використовуються іншими програмами кібербезпеки. Крім того, додавання нових експлойтів та зіставлень вразливостей є переважно ручним процесом, що ще більше ілюструє обмеження моделі в підтримці актуальності баз даних. Отже, моделювання нещодавно виявлених типів атак та нових вразливостей сервісів є складним завданням з урахуванням поточної інфраструктури експлойтів та зіставлень вразливостей.

Точні та детальні дані сповіщень датчиків IDS є дуже бажаними в сфері кібербезпеки для цілей тестування як інструментів обізнаності, так і інструментів реагування. Додаток для симуляції кібератак, розроблений під час цієї роботи, надає засоби для швидкого та ефективного створення репрезентативних даних сповіщень. Додаток також надає функціональність для маніпулювання представленнями віртуальної мережі, змодельованими кібератаками, виходами сповіщень датчиків та апріорними вхідними даними. Загалом, додаток можна розглядати як інструмент, який ефективно керує всім процесом моделювання сповіщень датчиків IDS. Хоча цей інструмент не є вичерпним у генерації даних сповіщень, він забезпечує достатню базову структуру, яку можна легко розширити та вдосконалити, щоб врахувати широкий спектр потенційних даних сповіщень.

Програма надає можливості мережевого моделювання, які дозволяють створювати чітко визначені мережеві представлення. Ці віртуальні мережі

включають роз'єми та машини/підмережі, з яких складаються фактичні мережі, а цим роз'ємам і машинам надаються відповідні атрибути. Наприклад, роз'єми включають дозволи брандмауера, які визначають, який тип мережевого зв'язку може відбуватися між певним джерелом і пунктом призначення в мережі. Також машини включають такі атрибути, як операційна система, запущені служби та тип машини, які використовуються для визначення конкретних вразливостей машини. Датчики IDS можуть бути розміщені як на роз'ємах, так і на машинах для належного моніторингу мережевого трафіку, що проходить через ці мережеві пристрої. Пристрої можуть бути підключені для формування певних, організованих рівнів у мережі, які вказують на глибину розташування машини в топології мережі (іншими словами , скільки з'єднань відділяє машину від зовнішньої частини мережі). Ця структура рівнів у поєднанні з атрибутами мережевого пристрою пропонує широкий спектр потенційних мережевих топологій, які можна створити за допомогою програми.

Додаток також надає можливості моделювання атак, які використовуються для створення детальних сценаріїв атак та імітації окремих дій атаки в рамках встановленої мережевої моделі. Крім того, виявлення таких дій датчиками створює сповіщення, якими можна маніпулювати та виводити як первинні результати моделювання. Функціональність для забезпечення таких можливостей включає моделювання мережевого трафіку, пов'язаного з діями атаки, зберігання набору дій атаки для кожної атаки та набору атак для кожного сценарію атаки, а також можливість генерувати атаки як вручну, так і автоматизовано та параметризовано. Цей автоматизований процес генерації атак використовує ряд параметрів, які надають користувачеві значний контроль над типом генерованої атаки та способом її розвитку в мережі. Цей процес також спирається на деталі мережевої структури, щоб забезпечити створення можливих прогресій атаки.

Функціональність мережевої моделі та симуляції атаки перевіряються та валідуються за допомогою різних підходів. Кілька функцій мережевого моделювання, включаючи структуру рівня конектора, реалізацію підмережі, представлення машинних служб та дозволи брандмауера конектора, перевіряються за

допомогою умовних операторів та тимчасового виводу налагодження. Інші мережеві функції візуально (або графічно) перевіряються шляхом спостереження за результуючою мережевою моделлю. Такі функції включають посилання з'єднання, IP-адреси, процес збереження та експорт віртуальної місцевості. Функції симуляції атаки перевіряються за допомогою унікального підходу для кожної функції. Правильна фільтрація експлойтів перевіряється шляхом спостереження за деталями окремих перелічених експлойтів. Етапи, отримані в процесі автоматизованої генерації атак, перевіряються шляхом спостереження за результатами сценаріїв та порівняння результатів із шаблоном інструкцій. Рівень контролю, що забезпечується параметрами атаки, перевіряється шляхом спостереження за тенденціями атак та іншими впливами на зміну параметрів. Маршрутизація сутності (трафіку) перевіряється шляхом відстеження відповідного шляху через мережеву модель. Генерація шуму перевіряється шляхом порівняння результатів шуму зі значеннями параметрів шуму. Нарешті, генерація сповіщень перевіряється шляхом перевірки виходу датчика на відповідність конкретним змодельованим діям атаки. Валідація процесу атаки та генерації сповіщень виконується шляхом оцінки можливостей моделювання з урахуванням того, як функціонує реальна мережа. Використання спільної сутності мережевого трафіку для представлення як атакуючих, так і шумових дій є дієвим способом представлення мережевого зв'язку та імітує реальну проблему неможливості чітко розрізнити дії, пов'язані з атакою, від типових шумових дій. Використання автоматизованих параметрів генерації атак дозволяє досвідченим користувачам легко створювати набір дійсних атак для задоволення конкретних потреб без необхідності визначення кожної конкретної дії. Крім того, вихідні дані сповіщень, отримані в результаті моделювання сценарію атаки, перевіряються як такі, що надають таку ж інформацію, як і реальні сповіщення відповідного типу датчика.

Поєднання моделі мережі та симуляції атаки надає загальній програмі широкий спектр можливостей. Деякі з ключових можливостей включають різноманітність конфігурацій мережі, які можна моделювати, взаємодію з віртуальною місцевістю, гнучкість шаблону введення, методологію, що надається для

генерації атак, моделювання шумової активності та генерацію як сповіщень датчиків IDS, так і реальних даних про сценарії атаки. Однак програма також містить деякі обмеження, які спонукають до подальшого розвитку та вдосконалення її функцій. Ці обмеження полягають у загальному представленні підключених пристроїв, відсутності набору типів датчиків IDS за замовчуванням, простоті та спостережуваній мінливості параметрів автоматизованої атаки, а також примітивних базах даних, що надаються для інформації про експлойти та вразливості.

Використання симулятора кібератак у співпраці з іншими програмами та суміжними галузями досліджень додатково демонструє можливості та корисність цієї програми. Взаємодія віртуальної місцевості, імітація сповіщень датчиків та вихідні дані про стан поверхні дозволяють програмі ефективно працювати з інструментом сканування мережі Nessus, інструментом об'єднання INFERD та інструментом оцінки VTAC. Функції моделювання мережі дозволяють імпортувати віртуальну місцевість, згенеровану з даними Nessus, як нову модель мережі та обробляти її так само, як модель мережі, згенеровану в програмі. Сповіщення датчиків, що генеруються симулятором, використовуються INFERD для кореляції дій атаки, і ці результати порівнюються з вихідними даними симулятора для оцінки продуктивності INFERD. Крім того, як віртуальна місцевість, так і вихідні дані сповіщень датчиків використовуються VTAC для визначення впливу певних дій на мережу.

У відповідь на початкове формулювання проблеми цієї роботи, розроблений додаток успішно відображає як розвиток атак через змодельовану мережу, так і генерацію точних сповіщень датчиків IDS в результаті таких атак. Моделі мережі та сценарії атак можуть бути створені та керовані з великою деталізацією. Крім того, сценарії атак можуть бути легко запущені та змінені. Сповіщення датчиків, що генеруються під час симуляції, є дійсними сповіщеннями датчиків IDS, пов'язаними з конкретними змодельованими діями атаки. Крім того, симулятор кібератак успішно інтегрується з іншими програмами та пов'язаними дослідженнями, щоб повною мірою використовувати функції моделювання мережі, симуляції атак та генерації сповіщень датчиків.

Симулятор кібератаки, розроблений у цій роботі, включає кілька досягнень, досягнутих у напрямку моделювання комп'ютерних мереж, сценаріїв кібератак та сповіщень систем виявлення вторгнень. Загалом, симулятор є кроком вперед у наданні репрезентативних даних про виявлення вторгнень; однак, сам по собі симулятор не виконує цього завдання. Сама програма має значні можливості як для вдосконалення існуючих функцій, так і для додавання нових. Крім того, вхідні дані, що використовуються симулятором, необхідно або реструктуризувати, або належним чином підтримувати, щоб надавати необхідну та актуальну інформацію, на яку спирається симулятор.

Щодо функціональності моделювання мережі, рекомендованим покращенням є використання більш специфічних типів роз'ємів. Існуюче представлення роз'єму (через клас роз'єму) є дуже узагальненим. Більш специфічні підкласи пристроїв підключення, такі як концентратори, маршрутизатори та комутатори, можна використовувати в моделі, просто розширивши клас роз'єму та успадкувавши атрибути та методи, передбачені для об'єктів роз'єму. У цій ситуації кожен конкретний клас роз'єму може містити додаткові методи та атрибути, що відображають конкретну функціональність типу роз'єму.

Датчики, змодельовані в мережевій структурі, також можна покращити, надаючи різним типам датчиків більш унікальні характеристики. Наприклад, фактичні мережеві та хост-сенсори IDS працюють та видають вихідні дані по-різному. Лише деякі відмінності між цими двома типами зображено в симуляції, а більшість реалізованих функцій датчиків представляють мережеві датчики IDS. Отримуючи як тип інформації, що виводиться хост-сенсором, так і визначення сповіщень, пов'язані з певною маркою хост-сенсора, мережу можна вдосконалити для точнішого відображення цих типів датчиків.

Що стосується функцій моделювання атак, основною рекомендованою функцією для додавання є можливість моделювання внутрішньої атаки. Хоча ця робота зосереджена на атаках зовнішнього характеру, внутрішні атаки є ще однією зростаючою загрозою для мережевої безпеки. Моделювання цього типу атаки наразі можливе лише в процесі ручного опису атаки. Щоб інтегрувати цей

тип атаки в процес автоматичної генерації атак, необхідно створити деякі додаткові атрибути автоматичної атаки та процеси вибору машини.

Процес автоматичної генерації атак можна додатково покращити, додавши нові параметри атаки. Хоча параметри ефективності, скритності та майстерності забезпечують значний рівень контролю з боку користувача, фактично згенеровані атаки все ще можуть бути досить непередбачуваними. Додаткові параметри можна створити, розбивши поточні три параметри на більш конкретні підпараметри, надавши додатковий параметр стандартного відхилення, пов'язаний з одним або кількома існуючими параметрами, або визначивши іншу особливість прогресії атаки, яку слід контролювати.

3.4 Висновок по розділу

У третьому розділі було розглянуто розробку, інтеграцію та оцінку ефективності програмного забезпечення для симуляції кібератак та тренування захисних команд. Реалізовано сценарії кібератак і тренувальні середовища, що дозволяють моделювати різні типи загроз та відпрацьовувати механізми реагування на інциденти. Досліджено процеси обробки подій, моніторингу активності та аналізу інцидентів безпеки для оцінювання дій користувачів у системі. Також проведено оцінку ефективності програмного забезпечення, проаналізовано результати експериментів та визначено перспективи подальшого розвитку системи. Отримані результати підтверджують доцільність використання кіберполігонів і систем симуляції для підвищення рівня підготовки фахівців із кібербезпеки.

ВИСНОВКИ

У ході виконання бакалаврської роботи було досліджено та розроблено програмне забезпечення для симуляції кібератак і тренування захисних команд, що є комплексним процесом, спрямованим на створення безпечного, масштабованого та ефективного середовища для підготовки фахівців із кібербезпеки. У роботі проаналізовано сучасні підходи до моделювання кіберінцидентів, досліджено методи побудови кіберполігонів і програмних платформ для тестування захисних механізмів, а також реалізовано функціональні можливості системи, які відповідають сучасним вимогам до інформаційної безпеки. Процес розробки охопив аналіз вимог, моделювання сценаріїв атак, реалізацію модулів обробки подій, тестування та оцінку ефективності програмного забезпечення, що забезпечило створення цілісного програмного рішення.

На початковому етапі було проведено аналіз предметної області та визначено функціональні й нефункціональні вимоги до системи. Основні функціональні можливості, такі як створення сценаріїв кібератак, моніторинг подій безпеки, аналіз дій користувачів та оцінка ефективності реагування захисних команд, були доповнені вимогами до масштабованості, продуктивності, надійності та безпеки даних.

У процесі моделювання бізнес-логіки було сформовано структуру системи та визначено взаємодію між її компонентами. Було розроблено архітектуру програмного забезпечення, що включає модулі генерації атак, аналізу подій, моніторингу активності та управління сценаріями навчання. Для опису структури та поведінки системи використовувалися UML-діаграми, зокрема Use Case, Sequence та Component Diagram, які дозволили формалізувати основні процеси та забезпечити ефективну реалізацію програмного рішення.

Реалізація системи охопила створення програмних модулів для запуску та моделювання кібератак, обробки журналів подій, взаємодії із системами моніторингу та візуалізації результатів роботи захисних команд. Особлива увага приділялася забезпеченню безпеки програмного середовища, ізоляції тестових

сценаріїв та захисту даних користувачів. Для цього були використані механізми автентифікації, журналювання дій, контролю доступу та обробки виняткових ситуацій. Це дозволило створити стабільне та безпечне середовище для проведення тренувань і тестування кіберзагроз.

Важливим етапом роботи стало тестування програмного забезпечення, яке включало модульне, інтеграційне, функціональне та навантажувальне тестування. Проведені експерименти підтвердили працездатність системи та її здатність ефективно моделювати сценарії атак і забезпечувати тренування команд реагування на кіберінциденти.

Загалом, розроблене програмне забезпечення відповідає поставленим цілям і дозволяє створювати реалістичні сценарії кібератак, проводити навчання захисних команд і здійснювати оцінку ефективності реагування на інциденти інформаційної безпеки. Перевагами запропонованого рішення є гнучкість архітектури, можливість масштабування, автоматизація процесів моделювання атак і підтримка інтеграції з сучасними засобами моніторингу безпеки. Разом із тим існують певні обмеження, зокрема необхідність подальшого вдосконалення механізмів генерації складних сценаріїв атак, підвищення рівня автоматизації аналізу подій та оптимізації продуктивності системи при високих навантаженнях.

У перспективі програмне забезпечення може бути розширене за рахунок інтеграції технологій штучного інтелекту для автоматичного виявлення аномалій, використання машинного навчання для прогнозування кіберзагроз, підтримки хмарних середовищ і створення адаптивних сценаріїв навчання. Це дозволить підвищити ефективність підготовки спеціалістів із кібербезпеки та сприятиме зміцненню захисту сучасних інформаційних систем від кіберзагроз.

СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА

1. Android Developers. (2025). Battery optimization for Android applications. developer.android.com. https://developer.android.com/topic/performance/power

2. Android Studio Profiler Documentation. (2025). developer.android.com. https://developer.android.com/studio/profile

3. Appium Mobile Testing Framework. (2025). appium.io. https://appium.io/docs/en/latest/

4. AWS Cybersecurity Simulation Solutions. (2025). aws.amazon.com. https://aws.amazon.com/security/

5. Bace, R., & Mell, P. (2001). Intrusion Detection Systems. National Institute of Standards and Technology. https://csrc.nist.gov/publications/detail/sp/800-31/archive/2001-11-28

6. Bishop, M. (2018). Computer Security: Art and Science. Addison-Wesley. https://www.pearson.com/en-us/subject-catalog/p/computer-security-art-and-science/P200000003094

7. Bloomfield, A., & Jones, K. (2024). Simulation environments for cybersecurity education. IEEE Access, 12, 45210–45228. https://doi.org/10.1109/ACCESS.2024.3378123

8. Burp Suite Documentation. (2025). portswigger.net. https://portswigger.net/burp/documentation

9. Cisco Cyber Range Platform. (2024). cisco.com. https://www.cisco.com/c/en/us/products/security/cyber-range.html

10. Clean Code Principles in .NET. (2023). devblogs.microsoft.com. https://devblogs.microsoft.com/dotnet/clean-code-principles/

11. Cyber Kill Chain Framework. (2024). lockheedmartin.com. https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html

12. Cypress End-to-End Testing. (2025). docs.cypress.io. https://docs.cypress.io/guides/overview/why-cypress

13. DARPA Cyber Grand Challenge Overview. (2024). darpa.mil. https://www.darpa.mil/program/cyber-grand-challenge

14. Elastic Stack Documentation. (2025). elastic.co. https://www.elastic.co/guide/index.html

15. Fowler, M. (2002). Patterns of Enterprise Application Architecture. martinfowler.com. https://martinfowler.com/eaaCatalog/

16. Freeman, E., & Robson, E. (2021). Head First Design Patterns. O'Reilly Media. https://www.oreilly.com/library/view/head-first-design/9781492077992/

17. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. https://www.pearson.com/store/p/design-patterns-elements-of-reusable-object-oriented/P100000252058

18. GitHub Security Lab. (2025). github.com. https://securitylab.github.com/

19. Hutchins, E., Cloppert, M., & Amin, R. (2011). Intelligence-Driven Computer Network Defense. Lockheed Martin. https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Intel-Driven-Defense.pdf

20. IBM Security QRadar Documentation. (2025). ibm.com. https://www.ibm.com/docs/en/qradar-common

21. Kali Linux Tools Documentation. (2025). kali.org. https://www.kali.org/tools/

22. Kim, D., & Solomon, M. (2023). Fundamentals of Information Systems Security. Jones & Bartlett Learning. https://www.jblearning.com/catalog/productdetails/9781284220735

23. Kumar, S., & Patel, R. (2024). Virtual cyber ranges for incident response training. Journal of Cybersecurity Technology, 8(2), 112–128. https://doi.org/10.1080/23742917.2024.2287641

24. Martin, R. C. (2017). Clean Architecture: A Craftsman's Guide to Software Structure and Design. pearson.com. https://www.pearson.com/store/p/clean-architecture-a-craftmans-guide-to-software-structure-and-design/P100001465201

25. Metasploit Framework Documentation. (2025). rapid7.com. https://docs.rapid7.com/metasploit/

26. Microsoft Defender for Cloud Documentation. (2025). learn.microsoft.com. https://learn.microsoft.com/en-us/azure/defender-for-cloud/

27. MITRE ATT&CK Framework. (2025). attack.mitre.org. https://attack.mitre.org/

28. MongoDB Documentation. (2025). mongodb.com. https://www.mongodb.com/docs/

29. National Institute of Standards and Technology. (2024). Cybersecurity Framework 2.0. csrc.nist.gov. https://www.nist.gov/cyberframework

30. .NET Documentation. (2025). learn.microsoft.com. https://learn.microsoft.com/en-us/dotnet/

31. OWASP Top Ten Security Risks. (2024). owasp.org. https://owasp.org/www-project-top-ten/

32. Postman API Platform Documentation. (2025). learning.postman.com. https://learning.postman.com/docs/

33. Scarfone, K., & Mell, P. (2007). Guide to Intrusion Detection and Prevention Systems (IDPS). NIST. https://csrc.nist.gov/publications/detail/sp/800-94/final

34. Selenium Documentation. (2025). selenium.dev. https://www.selenium.dev/documentation/

35. Snort Intrusion Detection System Documentation. (2025). snort.org. https://snort.org/documents

36. Splunk Security Essentials. (2025). splunk.com. https://www.splunk.com/en_us/software/splunk-security-essentials.html

37. Stallings, W. (2018). Network Security Essentials: Applications and Standards. Pearson Education. https://www.pearson.com/en-us/subject-catalog/p/network-security-essentials/P200000003479

38. VMware Cyber Range Solutions. (2024). vmware.com. https://www.vmware.com/topics/glossary/content/cyber-range.html

39. Wang, L., & Zhang, Y. (2025). AI-driven attack simulation frameworks for cyber defense training. ACM Transactions on Privacy and Security, 28(1), 1–22. https://doi.org/10.1145/3698821

40. Wireshark User Guide. (2025). wireshark.org. https://www.wireshark.org/docs/guide/

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи: " Розробка програмного забезпечення для симуляції кібератак та тренування захисних команд"

Обсяг пояснювальної записки: 90 аркушів

Дата закінчення бакалаврської роботи 10 червня 2026р.

Підпис студента _____