

Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра
на тему: «Проект підсистеми тестування захищеності
інформаційної системи з використанням Metasploit
Framework»

Виконав: студент 4-го курсу групи КБ-22-3
спеціальності 125 «Кібербезпека»

 Даніїл ШАХОВ

Керівник: Старший викладач кафедри КІ та КБ

 Ольга ДЯЧУК

Рецензент: Доцент кафедри КІ та КБ, к.т.н.

 Олександр ПІРОГ

Житомир – 2026

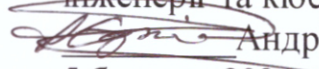
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

ЗАТВЕРДЖУЮ

Завідувач

кафедри комп'ютерної

інженерії та кібербезпеки

 Андрій ЄФІМЕНКО

5 березня 2026 р.

ЗАВДАННЯ

на випускню кваліфікаційну роботу бакалавра

Студент: Шахов Данііл Миколайович

Тема роботи: «Проект підсистеми тестування захищеності інформаційної системи з використанням Metasploit Framework»

затверджена Наказом університету від 2 лютого 2026 р. № 58/с

Термін здачі студентом закінченої роботи 22 червня 2026 р.

Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

Об'єкт дослідження: процес тестування захищеності інформаційних систем у контексті виявлення вразливостей, моделювання атак та оцінки рівня захисту корпоративної інфраструктури.

Предмет дослідження: методи, алгоритми та архітектурні рішення побудови інтегрованої підсистеми тестування захищеності інформаційної системи на основі Metasploit Framework як функціонального ядра, що забезпечує автоматизацію етапів пентесту, централізоване керування сценаріями атак та формування аналітичної звітності.

Вихідні дані для роботи:

1. Методологічна база: методології пентесту PTES та NIST SP 800-115 (базові), метрика критичності CVSS, база вразливостей CVE.
2. Модульна архітектура: scanner (збір даних), analyzer (аналіз вразливостей), exploit_engine (інтеграція з Metasploit), reporter (звітність), orchestrator (керуюче ядро) і CLI-інтерфейс.

3. Технологічний стек: Python 3.12; Metasploit Framework через msfrpcd (RPC API, MessagePack over HTTP); Nmap; БД SQLite; звіти у форматах HTML та JSON.
4. Середовище: ОС Linux (Kali Linux); ізольована мережа 192.168.10.0/24 на віртуальних машинах; цілі – Windows-інфраструктура (Server 2003 R2 / 2008 R2 / 2012 R2, Windows 7 Pro SP1).
5. Безпекові обмеження: whitelist дозволених підмереж, список заблокованих хостів, автентифікація до RPC API, журнал аудиту.

Обладнання та програмне забезпечення:

1. Робоча станція з підтримкою апаратної віртуалізації (Intel VT-x / AMD-V), ОЗП не менше 16 ГБ – для одночасного запуску віртуальних машин.
2. Хост-ОС: Linux/Windows.
3. Середовище виконання підсистеми – Linux (Kali Linux 2024.1 / Ubuntu 20.04+).
4. IDE PyCharm.
5. Python 3.12 – інтерпретатор та середовище виконання.
6. Metasploit Framework 6.4.
7. Nmap 7.94.
8. SQLite.

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Лайчук А.А.	02.02.2026 	 02.02.26
2	Лайчук А.А.	02.02.2026 	 02.02.26
3	Лайчук А.А.	02.02.2026 	 02.02.26

Студент  Данііл ШАХОВ

Керівник  Ольга ДЯЧУК

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання, вибір теми та затвердження плану	05.02 – 15.02	Виконано
2	Збір та аналіз літературних джерел, вивчення основ тестування на проникнення	16.02 – 10.03	Виконано
3	Аналіз існуючих інструментів (Metasploit, Nmap, OpenVAS) та методологій пентесту (PTES, NIST SP 800-115)	11.03 – 31.03	Виконано
4	Визначення вимог, проектування архітектури підсистеми та бази даних	01.04 – 15.04	Виконано
5	Розробка модуля мережевого сканування та збору даних (scanner.py)	16.04 – 30.04	Виконано
6	Розробка модуля аналізу вразливостей та інтеграції з Metasploit через RPC API	01.05 – 15.05	Виконано
7	Інтеграція модулів у єдину підсистему, розробка модуля звітності та CLI, тестування	16.05 – 22.05	Виконано
8	Експериментальне тестування на ізольованому стенді та оцінка ефективності	23.05 – 30.05	Виконано
9	Оформлення пояснювальної записки (тексту кваліфікаційної роботи)	31.05 – 05.06	Виконано
10	Підготовка презентації та демонстраційного матеріалу	05.06 – 08.06	Виконано
11	Захист	22.06	

Студент Д Даниїл ШАХОВ

Керівник О Ольга ДЯЧУК

АНОТАЦІЯ

Пояснювальна записка до випускної кваліфікаційної роботи бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. Загальний обсяг роботи – 100 сторінок, робота містить 9 рисунків, 3 лістинги, 14 таблиць та 1 додаток; список використаних джерел налічує 38 найменувань.

Роботу присвячено розробці підсистеми автоматизованого тестування захищеності інформаційної системи на базі Metasploit Framework. Актуальність теми зумовлена високою вартістю професійного тестування на проникнення та відсутністю серед наявних інструментів інтегрованого рішення з наскрізним автоматизованим робочим циклом.

У першому розділі проаналізовано загрози та моделі атак на інформаційні системи, методології тестування захищеності (PTES, NIST SP 800-115, OWASP, OSSTMM, ISSAF) і проведено порівняльний аналіз існуючих інструментів (Metasploit, Nmap, OpenVAS), що обґрунтував потребу у створенні інтегрованої підсистеми.

У другому розділі визначено функціональні, нефункціональні та безпекові вимоги, спроектовано модульну архітектуру підсистеми й механізм інтеграції Metasploit Framework як функціонального ядра через RPC API.

У третьому розділі реалізовано програмні модулі мовою Python 3.12 (збору даних, аналізу вразливостей, інтеграції з Metasploit та звітності), налаштовано взаємодію через msfrpcd і проведено експериментальне тестування на ізольованому стенді. Підтверджено 100 % точність виявлення вразливостей, успішність запуску exploit-модулів на рівні 61,9 %, а тривалість повного циклу тестування мережі класу /24 становить 75–150 секунд.

					ККІКБ.КР.Б – 125 – 10 – ПЗ		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Шахов Д.М.		02.06.26	Проект підсистеми тестування захищеності інформаційної системи з використанням Metasploit Framework		
Керівник		Дячук О.Ю.		10.06.26			
Реценз.		Пірог О.О.		10.06.26			
Н. контр.		Дячук О.Ю.		10.06.26			
Затверд.		Єфіменко А.А.		22.06.26			
					Лім.	Арк.	Аркушів
						5	100
					ФІКТ, Гр. КБ-22-3		

Об'єкт дослідження – процес тестування захищеності інформаційних систем у контексті виявлення вразливостей, моделювання атак та оцінки рівня захисту корпоративної інфраструктури.

Предмет дослідження – методи, алгоритми та архітектурні рішення побудови інтегрованої підсистеми тестування захищеності інформаційної системи на основі Metasploit Framework як функціонального ядра, що забезпечує автоматизацію етапів пентесту, централізоване керування сценаріями атак та формування аналітичної звітності.

Мета роботи – розробити та реалізувати підсистему тестування захищеності інформаційної системи, яка інтегрує Metasploit Framework через програмний інтерфейс і забезпечує автоматизоване проведення пентесту від збору розвідувальних даних до формування звіту про вразливості.

Практичне значення роботи полягає в можливості застосування підсистеми для зниження вартості та трудомісткості тестування захищеності, зокрема на малих і середніх підприємствах.

Ключові слова: ТЕСТУВАННЯ НА ПРОНИКНЕННЯ, METASPLOIT FRAMEWORK, АВТОМАТИЗАЦІЯ ПЕНТЕСТУ, ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ, МЕРЕЖЕВЕ СКАНУВАННЯ, RPC API, CVSS, ІНФОРМАЦІЙНА БЕЗПЕКА.

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

ANNOTATION

The explanatory note to the bachelor's qualification work consists of an introduction, three chapters, conclusions, a list of references, and appendices. The total volume of the work is 100 pages; it contains 9 figures, 3 listings, 14 tables, and 1 appendix; the list of references comprises 38 titles.

The work is devoted to the development of a subsystem for automated security testing of an information system based on the Metasploit Framework. The relevance of the topic stems from the high cost of professional penetration testing and the absence, among existing tools, of an integrated solution with an end-to-end automated workflow.

The first chapter analyses threats and attack models of information systems, security testing methodologies (PTES, NIST SP 800-115, OWASP, OSSTMM, ISSAF), and provides a comparative analysis of existing tools (Metasploit, Nmap, OpenVAS), which substantiates the need for an integrated subsystem.

The second chapter defines the functional, non-functional, and security requirements, designs the modular architecture of the subsystem, and develops the mechanism for integrating the Metasploit Framework as a functional core via the RPC API.

The third chapter implements the software modules in Python 3.12 (data collection, vulnerability analysis, Metasploit integration, and reporting), configures interaction through msfrpcd, and carries out experimental testing on an isolated test bench. A 100% accuracy of vulnerability detection was confirmed, the success rate of launching exploit modules reached 61.9%, and the duration of a full testing cycle for a /24 network was 75–150 seconds.

The object of the study is the process of security testing of information systems in the context of vulnerability detection, attack modelling, and assessment of the protection level of corporate infrastructure.

The subject of the study is the methods, algorithms, and architectural solutions for building an integrated subsystem for security testing of an information system

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

based on the Metasploit Framework as a functional core, which provides automation of the penetration testing stages, centralised management of attack scenarios, and the generation of analytical reporting.

The aim of the work is to develop and implement a subsystem for security testing of an information system that integrates the Metasploit Framework through a software interface and provides automated penetration testing – from the collection of reconnaissance data to the generation of a vulnerability report.

The practical value of the work lies in the possibility of applying the subsystem to reduce the cost and labour intensity of security testing, in particular for small and medium-sized enterprises.

Keywords: PENETRATION TESTING, METASPLOIT FRAMEWORK, PENTEST AUTOMATION, VULNERABILITY DETECTION, NETWORK SCANNING, RPC API, CVSS, INFORMATION SECURITY.

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

ІС – інформаційна система

ПЗ – програмне забезпечення

ОС – операційна система

API (Application Programming Interface) – програмний інтерфейс застосунку

CVSS (Common Vulnerability Scoring System) – система оцінювання критичності вразливостей

CVE (Common Vulnerabilities and Exposures) – база відомих вразливостей

RPC (Remote Procedure Call) – виклик віддалених процедур

HTTP (HyperText Transfer Protocol) – протокол передачі гіпертексту

HTTPS (HyperText Transfer Protocol Secure) – захищений протокол HTTP

TCP (Transmission Control Protocol) – протокол керування передачею

IP (Internet Protocol) – інтернет-протокол адресації

Nmap (Network Mapper) – інструмент для сканування мережі

MSF (Metasploit Framework) – фреймворк для тестування на проникнення

Exploit – програмний код для використання вразливості

Payload – корисне навантаження, що виконується після експлуатації

Post-exploitation – етап після отримання доступу до системи

RCE (Remote Code Execution) – віддалене виконання коду

DoS (Denial of Service) – відмова в обслуговуванні

DDoS (Distributed Denial of Service) – розподілена атака відмови в обслуговуванні

SQL (Structured Query Language) – мова запитів до баз даних

XML (eXtensible Markup Language) – мова розмітки даних

JSON (JavaScript Object Notation) – формат обміну даними

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ВСТУП	12
РОЗДІЛ 1. АНАЛІЗ ПІДСИСТЕМ ТЕСТУВАННЯ ЗАХИЩЕНОСТІ	15
1.1. Загрози та моделі атак інформаційних систем.....	15
1.2. Методи тестування захищеності	21
1.3. Аналіз існуючих рішень	24
1.4. Проблема відсутності інтегрованої підсистеми.....	32
1.5. Постановка задачі	37
Висновки до розділу 1	38
РОЗДІЛ 2. ПРОЄКТУВАННЯ ПІДСИСТЕМИ ТЕСТУВАННЯ.....	39
2.1. Вимоги до підсистеми	39
2.2. Архітектура підсистеми	41
2.3. Модель взаємодії компонентів	56
2.4. Алгоритми роботи підсистеми	61
2.5. Інтеграція Metasploit як ядра.....	66
Висновки до розділу 2	68
РОЗДІЛ 3 .РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПІДСИСТЕМИ.....	69
3.1. Реалізація програмних модулів	69
3.1.1. Модуль збору даних (scanner.py).....	70
3.1.2. Модуль аналізу вразливостей (analyzer.py).....	71
3.1.3. Модуль інтеграції з Metasploit (exploit_engine.py)	72
3.1.4. Модуль звітності (reporter.py).....	73
3.2. Налаштування Metasploit	74
3.2.1. Встановлення та конфігурація msfrpcd.....	74
3.2.2. Ініціалізація бази даних Metasploit.....	75
3.2.3. Конфігурація підсистеми для Windows-тестування.....	75
3.2.4. Генерація RC-скриптів для Metasploit	76
3.3. Реалізація сценаріїв тестування.....	77
3.3.1. Шестиетапний pipeline тестування	77

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

3.3.2. Сценарій тестування через CLI	78
3.4. Інтерфейс користувача	79
3.4.1. CLI-інтерфейс (main.py)	79
3.4.2. HTML-звіт.....	82
3.4.3. JSON-звіт.....	82
3.5. Експериментальне тестування.....	83
3.5.1. Тестовий стенд	83
3.5.2. Результати тестування виявлення вразливостей	84
3.5.3. Результати тестування модулів Metasploit	84
3.5.4. Вимірювання продуктивності.....	85
3.5.5. Тестування підсистеми безпеки	86
3.5.6. Висновки за результатами тестування.....	87
3.6. Обмеження системи та напрями вдосконалення	87
Висновки до розділу 3	89
ВИСНОВКИ.....	90
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	92
ДОДАТКИ.....	96
Додаток А. Лістинг main.py	96

ВСТУП

Стрімка цифровізація бізнес-процесів і зростання кількості кібератак на корпоративну інфраструктуру роблять проактивну оцінку захищеності інформаційних систем критично важливим завданням [32]. Одним із найефективніших методів такої оцінки є тестування на проникнення (penetration testing) – контрольована імітація реальної атаки з метою виявлення вразливостей до того, як ними скористаються зловмисники. Водночас професійний пентест є дорогою послугою, яка часто виходить за межі бюджету малих і середніх підприємств, а його якість значною мірою залежить від кваліфікації виконавця.

Наявні інструменти безпеки вирішують лише окремі ізольовані задачі: Nmap відповідає за мережеву розвідку, OpenVAS – за сканування вразливостей, Metasploit Framework – за експлуатацію. Між ними відсутня автоматизована передача даних, тому фахівець змушений вручну переносити результати з одного інструмента в інший, самостійно добирати exploit-модулі та складати звіти. Це підвищує вплив людського фактора, знижує відтворюваність тестів і збільшує їх трудомісткість. Відсутність інтегрованого рішення з наскрізним автоматизованим робочим циклом і централізованою звітністю обґрунтовує актуальність розробки відповідної підсистеми.

Мета роботи – розробити та реалізувати підсистему тестування захищеності інформаційної системи, яка інтегрує Metasploit Framework через програмний інтерфейс і забезпечує автоматизоване проведення пентесту – від збору розвідувальних даних до формування звіту про вразливості.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– проаналізувати актуальні загрози та моделі атак на інформаційні системи, дослідити методології тестування захищеності (PTES, NIST SP 800-115, OWASP, OSSTMM, ISSAF) та провести порівняльний аналіз наявних інструментів (Metasploit, Nmap, OpenVAS);

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

- обґрунтувати відсутність серед існуючих інструментів інтегрованого автоматизованого рішення та сформулювати проблему дослідження;
- визначити функціональні, нефункціональні та безпекові вимоги до підсистеми;
- розробити модульну архітектуру підсистеми з виділенням компонентів збору даних, запуску атак, аналізу та звітності, побудувати модель взаємодії компонентів;
- спроектувати та реалізувати механізм інтеграції Metasploit Framework як функціонального ядра через RPC API;
- реалізувати програмні модулі, сценарії тестування та інтерфейс користувача з підтримкою формування звітів;
- провести експериментальне тестування підсистеми на ізольованій мережі та оцінити її ефективність за критеріями точності, продуктивності й ступеня автоматизації.

Об'єкт дослідження – процес тестування захищеності інформаційних систем у контексті виявлення вразливостей, моделювання атак та оцінки рівня захисту корпоративної інфраструктури.

Предмет дослідження – методи, алгоритми та архітектурні рішення побудови інтегрованої підсистеми тестування захищеності інформаційної системи на основі Metasploit Framework як функціонального ядра, що забезпечує автоматизацію етапів пентесту, централізоване керування сценаріями атак та формування аналітичної звітності.

Методи дослідження. У роботі використано: системний та порівняльний аналіз – для дослідження загроз, методологій та наявних інструментів; методи об'єктно-орієнтованого проектування і UML-моделювання – для побудови архітектури та моделі взаємодії компонентів; методи структурного й об'єктно-орієнтованого програмування мовою Python – для реалізації модулів; методології тестування на проникнення (PTES, NIST SP 800-115) – для

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

побудови сценаріїв пентесту; експериментальні методи та кількісну оцінку результатів – для перевірки ефективності підсистеми на тестовому стенді.

Практичне значення одержаних результатів полягає в тому, що розроблена підсистема дає змогу автоматизувати повний цикл тестування захищеності, знизити його вартість і трудомісткість та зробити пентест доступнішим для малих і середніх підприємств. Підсистему може бути використано системними адміністраторами, фахівцями з кібербезпеки, а також у навчальному процесі для підготовки спеціалістів зі спеціальності 125 «Кібербезпека». Модульна архітектура забезпечує можливість подальшого розширення функціоналу без переробки ядра.

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ПІДСИСТЕМ ТЕСТУВАННЯ ЗАХИЩЕНОСТІ

1.1. Загрози та моделі атак інформаційних систем

Інформація – це складне та абстрактне поняття, яке може бути виражене в різних формах, таких як діаграми, дані, символи, числа, програми, відео тощо. Різні форми інформації мають різні формати, кодування, синтаксис та інші структурні типи. Інформація – це контент, що передається під час процесу перетворення з однієї форми або стану в інший. Щоб забезпечити безпеку інформації, важливо спочатку зрозуміти її характеристики, а потім вжити цілеспрямованих заходів безпеки [19].

В інформатиці інформація зазвичай представлена у вигляді чисел або символів і передається та обробляється через комп'ютерні мережі, носії інформації або інші методи передачі. Інформація може бути використана для опису стану, атрибутів, зв'язків та поведінки речей і може підтримувати прийняття рішень, комунікацію, контроль та інші види діяльності.

Поняття інформації має важливе значення в багатьох галузях, включаючи комунікації, статистику, теорію інформації, інформатику, штучний інтелект, філософію та інші. Тому необхідно узагальнити характеристики різних типів інформації та вжити відповідних заходів безпеки, щоб запобігти знищенню форми, типу та інших аспектів інформації.

Надійність та цілісність інформаційної безпеки вимагають численних заходів контролю та захисних систем для збереження початкової якості інформації. Щоб запобігти знищенню інформації, необхідно вжити відповідних заходів маскування або надати спеціальні дозволи персоналу чи пристроям служби безпеки. Це важливий крок для забезпечення цілісності інформації.

Мережева інформаційна безпека – це механізм безпеки, створений для того, щоб гарантувати, що інформація не буде незаконно вторгнута, знищена або викрадена під час передачі мережею. За допомогою різних методів захисту

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

безпеки та заходів управління мережеві пристрої, програми та інформація захищені для запобігання зловмисним атакам та забезпечення безпечної роботи мережевих інформаційних систем. Крім того, інформаційна безпека також захищає інформацію та дані в комп'ютерних системах, мережах, мобільних пристроях та інших цифрових пристроях, запобігаючи несанкціонованому доступу, використанню, знищенню, підробці, розголошенню та злому. Інформаційна безпека є продуктом розвитку комп'ютерних технологій і є невід'ємною частиною сучасного суспільства. Сфера інформаційної безпеки включає такі аспекти, як безпека комп'ютерних мереж, безпека даних, безпека програм, безпека операційних систем та фізична безпека [26].

Метою інформаційної безпеки є забезпечення конфіденційності, цілісності та доступності інформації. Конфіденційність означає, що доступ до інформації можуть мати лише авторизовані користувачі або системи, а не неавторизовані користувачі чи системи. Цілісність означає, що інформація не може бути змінена або знищена неавторизованими користувачами, що забезпечує точність та надійність інформації. Доступність означає, що інформація має бути доступною за потреби та не може бути незаконно перехоплена або заблокована, що забезпечує своєчасність та доступність інформації. Захист цих трьох аспектів є основою інформаційної безпеки та допомагає організаціям та окремим особам захищати свою критично важливу інформацію від загроз та атак. Для досягнення інформаційної безпеки необхідно вживати різних заходів безпеки, включаючи автентифікацію, контроль доступу, шифрування, брандмауери, виявлення та запобігання вторгненням тощо. Крім того, необхідні навчання та освіта з питань безпеки для підвищення обізнаності користувачів щодо безпеки, запобігання витокам інформації та інцидентам безпеки, спричиненим людською помилкою.

Інформаційна безпека – це галузь, що постійно розвивається та змінюється. З появою та постійним удосконаленням нових технологій інформаційна безпека стикається з новими викликами та загрозами. Тому

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

підтримка інформаційної безпеки вимагає постійних досліджень та вдосконалень для боротьби з постійно мінливими загрозами.

Безпека мережевої інформаційної системи стосується використання технічних та управлінських заходів для захисту комп'ютерних мереж та інформаційних систем від різних загроз, таких як несанкціонований доступ, використання, розкриття, знищення та стороннє втручання. Це критично важливий гарант надійності, конфіденційності, цілісності, доступності та керованості мережевих та інформаційних систем, що стосується значних інтересів, таких як національна безпека, економічна безпека, соціальна стабільність та особиста недоторканність. Її забезпечення потребує узгодженої взаємодії технологій, управління та нормативно-правового регулювання. Безпека є важливою гарантією для мережевих інформаційних систем, і будь-яка вразливість в одній частині може загрожувати безпеці всієї системи. Тому захист слід впроваджувати комплексно на всіх етапах, включаючи проектування, розробку, розгортання та обслуговування системи. Водночас, передові технічні заходи, такі як брандмауери, системи виявлення вторгнень та технології шифрування, можуть покращити безпеку системи [18].

Наразі не існує стандартного визначення тестування на проникнення. Зазвичай його описують як симульовану атаку на певну мережу або програму в реальному світі, що включає різні варіанти та функції. Тестування на проникнення зазвичай виконує професійний тестер на проникнення або аудитор безпеки. З технічної точки зору, тестування на проникнення – це комплексна перевірка безпеки системи, як внутрішньої, так і зовнішньої, для виявлення потенційних вразливостей, які можуть використати зловмисники. Іншими словами, воно включає оцінку компонентів архітектури мережевої безпеки, включаючи операційні системи, засоби зв'язку, програми, мережеві пристрої, фізичну безпеку та людську психологію.

Простим прикладом тестування на проникнення є використання пошукових систем для тестування. У своїй книзі **Google Hacking for Penetration Testers** Джонні Лонг ділиться багатьма порадами, такими як те, як

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

використовувати величезну базу даних Google для отримання корисної інформації для фахівців з кібербезпеки та тестувальників на проникнення; як використовувати підказки для пошуку інформації, пов'язаної з цільовою аудиторією (наприклад, контактні дані співробітників, адреси електронної пошти тощо); як ідентифікувати вразливе програмне забезпечення, як створити карту веб-сторінки тощо. Крім того, Google може швидко створити список найбільш вразливих серверів у всьому світі, надаючи корисну інформацію для навчених хакерів.

Для забезпечення безпеки мережі тестування на проникнення вимагає як практичної експлуатації [3], так і загального планування та проектування. Типовий процес тестування на проникнення включає кілька кроків: сканування IP-адрес, виявлення вразливих серверів, виявлення непатчених операційних систем, запис результатів тестування та написання звітів для подання. Протягом усього процесу тестування на проникнення використовуються різні ручні та автоматизовані методи для проведення комплексного аналізу системи, виявлення її вразливостей та надання корисної інформації користувачам. Якщо їх виконати неправильно, це може призвести до перевантаження мережі, збоїв системи, сповіщень про виявлення вторгнень або навіть вимкнення пристроїв.

Огляд Класифікація загроз важлива, оскільки вона головним чином дозволяє ідентифікувати та розуміти загрози, їх характеристики та джерела для захисту системних активів. Крім того, вона визначає ризики безпеки, що загрожують цим системам, та допомагає у розумінні можливостей та виборі рішень безпеки [37]. Загроза – це мета зловмисника або те, що зловмисник може спробувати зробити з системою [7]. Вона також описується як здатність зловмисника атакувати систему [7]. Таким чином, загрозу можна визначити двома способами: методи, які зловмисники використовують для використання вразливостей у компонентах вашої системи, або вплив загроз на ваші активи. Отже, ми можемо розділити підходи до класифікації загроз на два основні

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

класи [46]: Методи класифікації, що базуються на методах атак та Методи класифікації, що базуються на впливі загроз

Таблиця 1.1

Порівняння моделей

Модель	Джерело	Основа класифікації	Критерії / виміри	Особливості
Методи класифікації на основі методів атак				
Тривимір на модель	Академічна	Методи атак та суб'єкти загроз	- Агент - Мотивація - Локалізація	Ортогональні виміри; поділяє простір загроз на підпростори. Агент: людський / технологічний / форс-мажор
Кубічна модель СЗ	Академічна	Гібридний підхід	- Частота загроз - Область активності - Джерело загрози	Охоплює фізичну, операційну, комунікаційну безпеку та безпеку персоналу
Піраміда IST	Академічна	Навмисні загрози	- Знання зловмисника - Критичність зони - Втрати	Зосереджена виключно на навмисних атаках; враховує обізнаність атакуючого про систему
Методи класифікації на основі впливу загроз				
STRIDE	Промислова	Цілі та мотивація атак	- Підробка - Фальсифікація - Відмова - Розкриття DoS - Ескалація	Застосовується до мережі, хоста та застосунку; цілеспрямований підхід – оцінка з позиції зловмисника
Модель ISO (ISO 7498-2)	Стандарт	Вплив на інформацію та ресурси	- Знищення - Пошкодження - Крадіжка - Розкриття - Переривання	Міжнародний стандарт; еталонна модель для опису загроз; базується на ЦДД (конфіденційність, цілісність, доступність)

Методи класифікації на основі методів атак триортогональна вимірна модель Лукас Руф та ін. запропонували в [36] нову модель загроз для

категоризації загроз безпеці з метою покращення розуміння загроз та полегшення існуючих моделей класифікації загроз.

Вона вирішує цю проблему шляхом введення тривимірної моделі, яка поділяє простір загроз на підпростори відповідно до трьох ортогональних вимірів, позначених як мотивація, локалізація та агент:

- Агент загрози – це суб'єкт, який нав'язує загрозу певному активу системи, який представлений трьома класами: людський, технологічний та форс-мажор.

- Мотивація загрози представляє причину створення загрози та реорганізується у два класи: навмисні та випадкові загрози

- Локалізація загрози представляє походження загроз, внутрішніх або зовнішніх.

Гібридна модель для класифікації загроз. Кубічна модель (С3) класифікує загрози за трьома основними критеріями:

- Частота загроз безпеці – показує частоту виникнення загроз безпеці.

- Область активності загроз безпеці – представляє область, на яку впливає загроза, таку як фізична безпека, безпека персоналу, безпека зв'язку та даних, а також операційна безпека.

- Джерело загрози безпеці – надає типи джерела загрози.

Мохаммед Альхабіб та ін. у [34] пропонують гібридний метод класифікації навмисних загроз безпеці, названий «Пірамідою класифікації загроз в інформаційній безпеці». Він класифікує навмисні загрози за трьома факторами:

- попередня обізнаність зловмисника — наскільки він знає систему: її апаратне й програмне забезпечення, а також персонал і користувачів;

- критичність зони ураження — важливість частин системи, на які здатна вплинути загроза;

- потенційні втрати — можливі збитки для системи чи організації (конфіденційність, цілісність тощо).

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

STRIDE — це модель, що дає змогу класифікувати відомі загрози за цілями та мотивацією атак. Її назва є акронімом шести категорій загроз:

- S (Spoofing) — підробка ідентичності;
- T (Tampering) — фальсифікація даних;
- R (Repudiation) — невизнання (заперечення) дій;
- I (Information disclosure) — розкриття інформації;
- D (Denial of service) — відмова в обслуговуванні;
- E (Elevation of privilege) — підвищення привілеїв.

Підхід є цілеспрямованим: він дає змогу оцінити систему з позиції зловмисника та виявити характерні загрози для кожної категорії.

Еталонна модель ISO 7498-2 виокремлює п'ять основних категорій загроз безпеці [35]:

- знищення інформації та/або інших ресурсів;
- пошкодження або модифікація інформації;
- крадіжка, видалення чи втрата інформації та/або інших ресурсів;
- розкриття інформації;
- переривання обслуговування (надання послуг).

1.2. Методи тестування захищеності

Методологія тестування захищеності визначає структурований підхід до проведення пентесту: послідовність етапів, правила взаємодії (Rules of Engagement, RoE), критерії оцінки та вимоги до звітності. Вибір методології безпосередньо впливає на повноту, відтворюваність та юридичну коректність тестування. У цій частині розглянуто та порівняно п'ять провідних методологій:

- OWASP Testing Guide,
- PTES,
- OSSTMM,
- NIST SP 800-115 та ISSAF.

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

Проект безпеки відкритих веб-додатків (OWASP) зосереджений на тестуванні безпеки веб-застосунків та API. OWASP Testing Guide надає детальні тест-кейси для більш ніж 100 категорій вразливостей, класифікованих за векторами атак та бізнес-ризиками. Методологія охоплює збір інформації, управління конфігурацією, тестування автентифікації та авторизації, управління сесіями, бізнес-логіку та ін'єкції. Регулярно оновлюється великою відкритою спільнотою. Обмеження – вузька область застосування (лише web), без покриття мережевої інфраструктури [5].

Стандарт виконання тестування на проникнення (PTES, 2009) [16] встановлює єдиний framework для повного циклу пентесту. Він поділений на сім фаз: вимоги до взаємодії, збір розвідувальних даних, моделювання загроз, аналіз вразливостей, експлуатація, пост-експлуатація та звітність. Кожна фаза детально описує кроки, інструменти та критерії завершення. PTES є найбільш структурованою методологією для корпоративної інфраструктури та слугує основою для побудови автоматизованого pipeline підсистеми у цій роботі.

Посібник з методології тестування безпеки з відкритим кодом (OSSTMM) виділяється серед інших завдяки кількісному підходу: результатом тестування є чисельна метрика Attack Surface (RAV), а не лише перелік вразливостей. Методологія поділяє тестування на канали (мережевий, фізичний, людський, бездротовий) та визначає чіткі правила взаємодії (RoE). OSSTMM є оптимальним для аудиту безпеки та нормативної відповідності, проте складний у реалізації та потребує значних ресурсів.

NIST SP 800-115 «Технічний посібник з тестування та оцінки інформаційної безпеки» розроблений Національним інститутом стандартів і технологій США. Документ деталізує ролі учасників тестування, вимоги до правил взаємодії та порядок документування результатів. Особливістю є чітке розмежування між оцінкою вразливостей та повноцінним тестуванням на проникнення. Принципи управління ролями та RoE з NIST 800-115 використані при проектуванні модуля SecurityValidator у даній роботі.

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

Структура оцінки безпеки інформаційних систем (ISSAF) надає детальні технічні кроки для тестування конкретних компонентів: брандмауерів, IDS, маршрутизаторів, VPN, веб-серверів та баз даних. Методологія включає три фази: планування та підготовку, оцінку, звітність та виправлення. Перевагою є практична деталізованість, що базується на реальних кейсах. Основний недолік – документ не оновлювався з 2006 року. [38].

Таблиця 1.2

Порівняння методологій тестування захищеності

Методологія	Охоплення / сфера застосування	Переваги	Недоліки	Рішення щодо застосування
OWASP Testing Guide	Веб-додатки та API	Детальні тест-кейси, велика спільнота, регулярні оновлення	Лише web; не охоплює мережеву інфраструктуру	✗ Не обрано: проєкт орієнтований на мережеву інфраструктуру
PTES	Мережі, системи, застосунки; корпоративна інфраструктура	Структурований процес із семи фаз (від розвідки до звіту); охоплює повний цикл пентесту	Частина описів застаріла; бракує технічних деталей	✓ Обрано як основу: фази PTES відповідають архітектурі підсистеми
OSSTMM	Мережі, фізична безпека, телекомунікації; аудит, нормативна відповідність	Кількісні метрики (RAV), правила взаємодії (RoE), висока повторюваність	Складна реалізація; значний обсяг документації	✗ Не обрано: надлишково складна для прототипу
NIST SP 800-115	ІТ-системи, держсектор, критична інфраструктура	Чіткі ролі учасників, деталізовані RoE, визнаний держстандарт США	Менш деталізований технічно; орієнтований на процеси, а не на інструменти	● Частково використано: принципи RoE та управління ролями в SecurityValidator
ISSAF	Мережі, системи, ПЗ; технічне тестування, аудит	Детальні технічні кроки, базується на багаторічній практиці	Не оновлювався з 2006 р.; може бути застарілим	✗ Не обрано: документ застарілий

Таким чином, для реалізації підсистеми обрано комбінований підхід: на основі семифазної PTES у підсистемі реалізовано спрощений шестиетапний pipeline, а принципи правил взаємодії та управління ролями запозичені з NIST

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

SP 800-115. Такий вибір забезпечує повноту охоплення етапів пентесту та коректність з точки зору інформаційної безпеки [7].

1.3. Аналіз існуючих рішень

Metasploit – це фреймворк з відкритим кодом для тестування на проникнення, який допомагає дослідникам безпеки, тестувальникам на проникнення та аналітикам вразливостей виявляти та використовувати вразливості, а також оцінювати ефективність захисту мережевої безпеки. Створений Н. D. Moore у 2003 році, Metasploit став одним із найпопулярніших та найширше використовуваних фреймворків у галузі тестування на проникнення [1].

Фреймворк Metasploit складається з трьох основних компонентів: Metasploit Framework, Metasploit Pro та Metasploit Community Edition. Metasploit Framework є базовим і найбільш широко використовуваним компонентом, що пропонує комплексний набір інструментів та модулів для таких завдань, як сканування вразливостей, експлуатація, генерація корисного навантаження, розробка шелл-коду та пост-експлуатація.

Основні компоненти Metasploit Framework викладено нижче [1]:

- бібліотека модулів – репозиторій, що містить модулі для сканування вразливостей, експлуатації, генерації корисного навантаження, написання шелл-коду та постексплуатації;
- інтерфейси – інтерфейс командного рядка (CLI), графічний інтерфейс (GUI) через Armitage та веб-інтерфейси;
- інтеграція з базами даних – офіційна підтримка PostgreSQL для зберігання та керування даними про хости, сервіси й вразливості;
- корисні навантаження – програмні модулі, що виконують дії на цільових системах: отримання доступу до оболонки, завантаження/вивантаження файлів, виконання команд;

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

- експлойти – модулі для використання системних вразливостей з метою отримання доступу до оболонки;
- допоміжні модулі – засоби для допоміжних завдань: сканування портів, ідентифікації сервісів, перевірки вразливостей.

Фреймворк Metasploit є дуже гнучким, що дозволяє користувачам вибирати та налаштовувати різні модулі та експлойти, адаптовані до конкретних цілей. Крім того, користувачі можуть розробляти власні модулі та експлойти для конкретних сценаріїв тестування, що підвищує адаптивність інструменту до різноманітних середовищ та проблем безпеки [1].

Перш ніж розпочати атаку за допомогою Metasploit, тестувальники на проникнення повинні спочатку зібрати детальну інформацію про ціль. Metasploit містить величезну базу даних вразливостей, пов'язаних із програмами, протоколами, сервісами та операційними системами. Після виявлення вразливості тестувальники вибирають відповідний модуль експлойту. У разі успіху експлойт дозволяє тестувальнику виконати модуль корисного навантаження на цільовій машині. Ці корисні навантаження можуть повертати оболонку або встановлювати з'єднання між цільовою машиною та машиною тестувальника, тим самим забезпечуючи віддалене виконання команд та подальшу взаємодію з системою.

Перед виконанням експлойту корисне навантаження має бути належним чином попередньо налаштоване. Щоб уникнути виявлення системами запобігання та виявлення вторгнень, корисне навантаження можна закодувати. Metasploit підтримує автоматизацію всього процесу експлуатації за допомогою відповідних сценаріїв та налаштувань, оптимізуючи робочі процеси та підвищуючи ефективність тестування. Підсумовуючи, Metasploit Framework – це потужний, зручний у використанні та розширюваний інструмент для тестування на проникнення – важлива платформа для дослідження вразливостей, експлуатації та тестування безпеки. На рис. 1.1 ілюструється архітектура Metasploit [1].

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				25
Змн.	Арк.	№ докум.	Підпис	Дата		

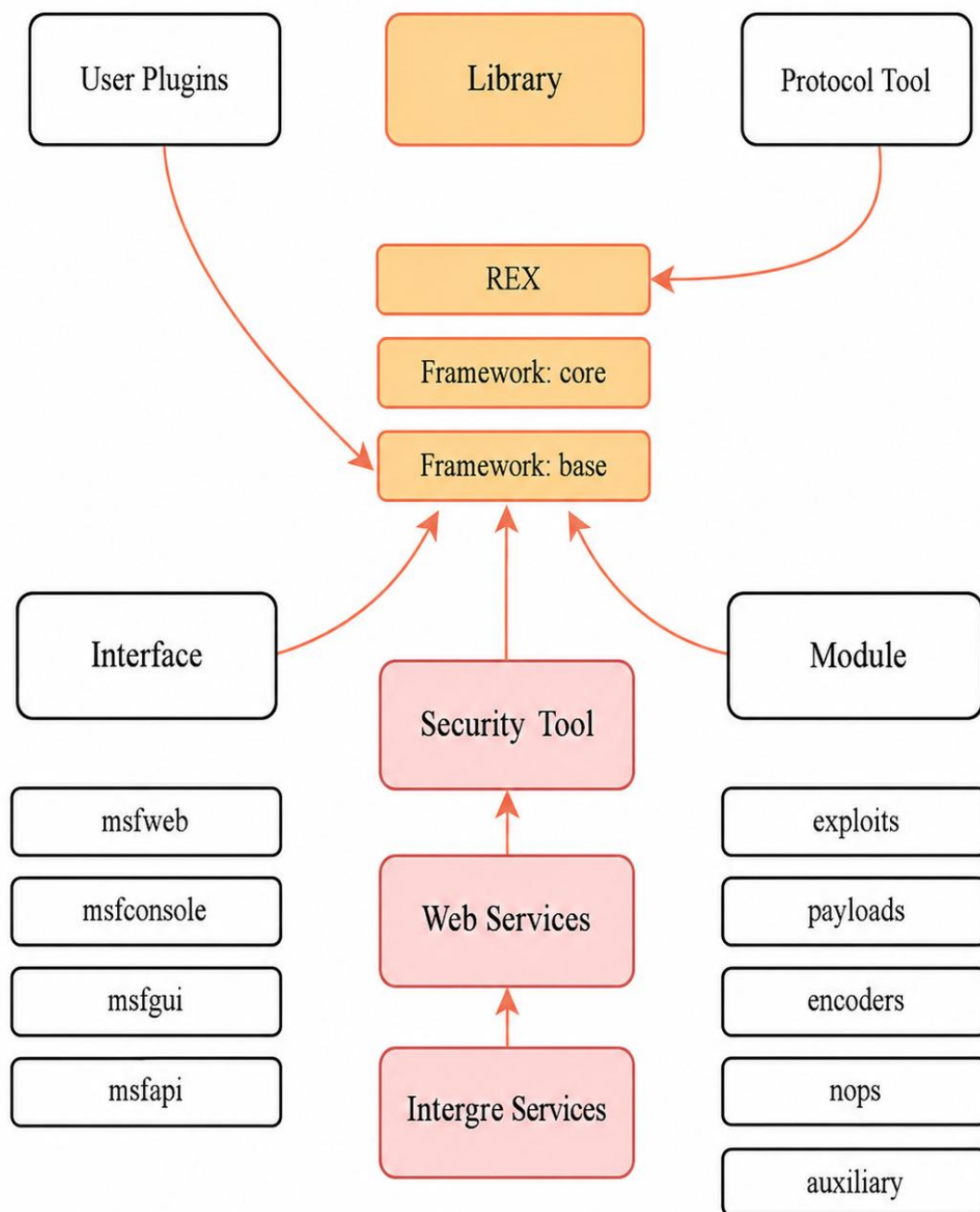


Рисунок 1.1 – Архітектура Metasploit

Metasploit містить різні підкаталоги для взаємодії з основним фреймворком, які поділяються на інтерфейси, модулі, бібліотеки, плагіни та інструменти. REX – це розширена бібліотека Ruby, яка інтегрує класи та методи в базові файли Metasploit. Інтерфейси пропонують п'ять основних опцій: msfweb, msfcli, msfconsole, msfgui та msfapi, серед яких msfconsole є найпотужнішим, надаючи користувачам повний доступ до всіх функцій

Metasploit. Крім того, інтерфейс плагінів дозволяє розширювати архітектуру та можливості фреймворку. Справжня сила Metasploit значною мірою полягає в його модульності, оскільки кожен модуль виконує окрему та конкретну функцію в процесі тестування на проникнення [9].

Nmap – це безкоштовний програмний інструмент, розроблений Fyodor та випущений за ліцензією GPL 2.0. Завдяки своєму відкритому коду, він здобув широку популярність, особливо серед фахівців з тестування на проникнення. Його гнучкість, стабільність, сумісність та масштабованість відрізняють його від інших інструментів сканування. Крім того, завдяки чудовій підтримці та потужним можливостям підтримки хостів, Nmap служить ефективним інструментом для сканування мережі та моніторингу хостів [5].

Основні функції Nmap включають сканування хостів, служб, операційних систем, фільтрів пакетів та брандмауерів. Він має розширені можливості, які дозволяють йому обходити брандмауери та системи виявлення вторгнень. Крім того, Nmap пропонує численні приховані функції, які можна інтегрувати в скрипти та інші програми. Поряд з цим, він також постачається з набором допоміжних інструментів, наприклад, Ncat для налагодження, Ndiff для порівняння результатів сканування та Nping для генерації пакетів та аналізу відповідей, серед інших.

Щодо виводу, Nmap представляє результати сканування у форматі списку на основі вибраних опцій. Наприклад, список портів відображає номер порту, протокол, назву служби та стан, зі статусами, що включають відкритий, фільтрований, закритий, нефільтрований та невизначений. Зокрема, «відкритий» вказує на те, що Nmap виявив службу на цільовому хості, тоді як «фільтрований» означає, що брандмауер або інший мережевий бар'єр блокує порт, запобігаючи отриманню відповіді Nmap. Крім того, механізм сценаріїв Nmap (NSE) – це потужний фреймворк, призначений для виконання широкого спектру завдань, пов'язаних з мережею, включаючи виявлення мережі, розширене виявлення версій, сканування вразливостей, виявлення бекдорів та

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				27
Змн.	Арк.	№ докум.	Підпис	Дата		

навіть експлуатацію вразливостей. Користувачі можуть використовувати NSE для автоматизації різних операцій, які виконуються паралельно, оптимізуючи продуктивність та швидкість. На додаток до вбудованих скриптів, користувачі можуть розробляти власні скрипти, адаптовані до конкретних вимог. Ці власні скрипти особливо цінні в тестуванні на проникнення, дозволяючи проводити більш цілеспрямовані та ефективні оцінки [2].

OpenVAS, також відомий як GNessus, розроблений на базі Nessus і слугує комплексною та потужною платформою для сканування та управління вразливостями. Як частина рішення Greenbone Vulnerability Management (GVM), OpenVAS виник як форк Nessus у 2005–2006 роках. Сканер OpenVAS регулярно оновлюється за допомогою тестів мережевих вразливостей (NVT) із загальнодоступних джерел. Кількість NVT перевищила 50 000 і продовжує зростати. Усі компоненти OpenVAS є вільно доступними, більшість з них ліцензовані за Загальною публічною ліцензією GNU (GPL).

Ключовою особливістю OpenVAS є його здатність використовувати плагіни для ефективного зіставлення та виявлення вразливостей [15].

Архітектуру OpenVAS можна розділити на три основні рівні: клієнтський рівень, рівень сервісів та рівень даних. Основні компоненти та модулі в цих рівнях проілюстровано на рис.1.2.

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

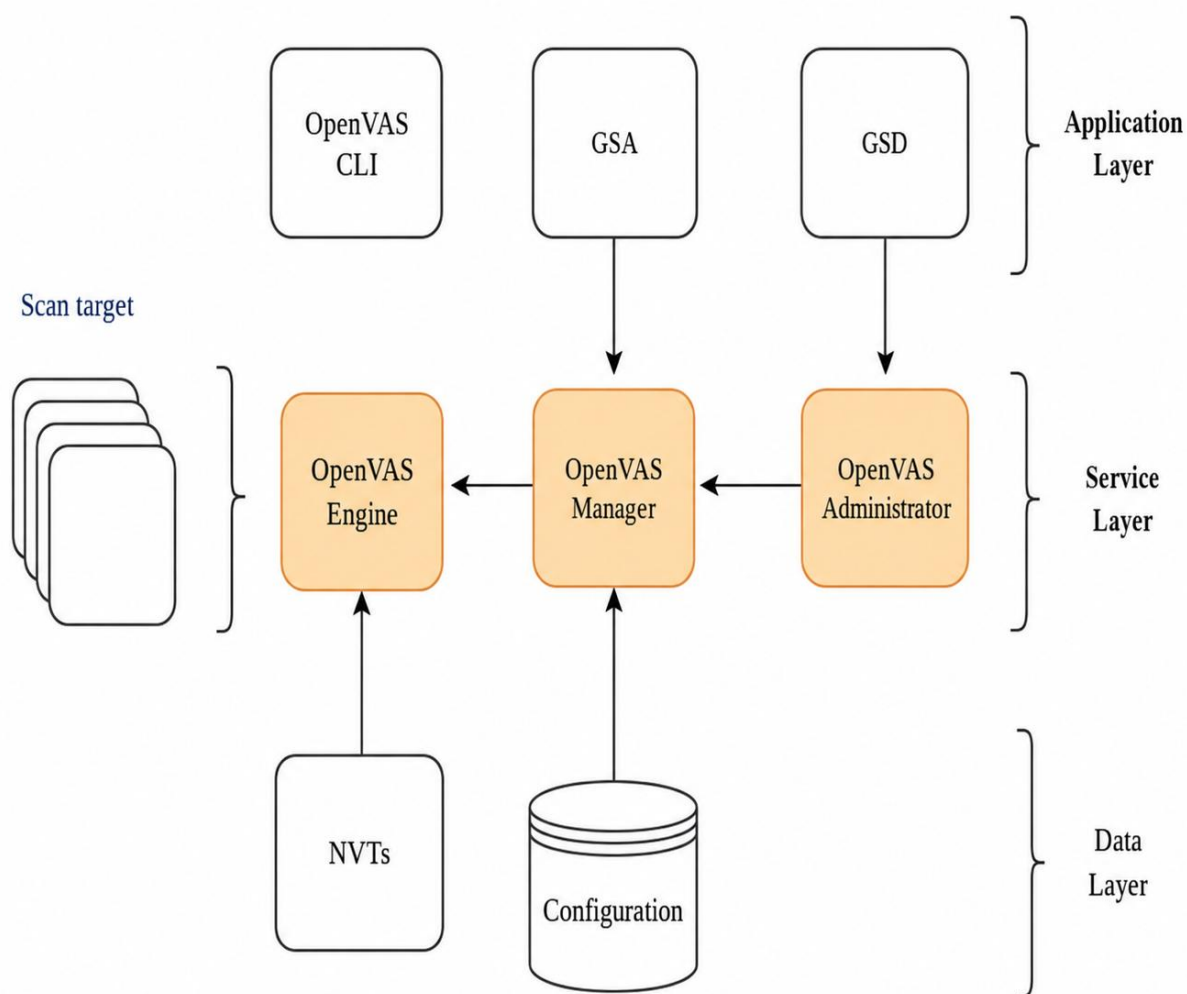


Рисунок 1.2 – Три рівні OpenVAS

VMware Workstation – це програмний гіпервізор, який дозволяє створювати та запускати декілька віртуальних машин на одному фізичному комп’ютері в межах хост-операційної системи. Платформа постійно розвивається та отримує нові функціональні можливості, спрямовані на підвищення продуктивності, стабільності та зручності керування віртуальним середовищем. Інтерфейс VMware Workstation реалізований у вигляді графічного середовища керування та доповнюється засобами командного рядка для автоматизації операцій з віртуальними машинами.

Для забезпечення тісної інтеграції між хостовою та гостьовою операційними системами VMware надає пакет VMware Tools, який включає підтримку спільних папок, синхронізацію буфера обміну, покращену обробку

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

графіки, автоматичне налаштування драйверів, синхронізацію часу та можливість безперешкодного переміщення курсора між віртуальною та хостовою системами. Ці можливості значно спрощують роботу користувача та підвищують ефективність використання віртуального середовища.

Таблиця 1.3

Порівняльна характеристика інструментарію

Інструмент	Тип / ліцензія	Основне призначення	Ключові модулі	Інтерфейси	Обмеження
Metasploit Framework	Open Source, BSD	Експлуатація вразливостей, постексплуатація, генерація payload	Exploit, Payload, Auxiliary, Post, Encoder	msfconsole, msfapi, Armitage GUI, msfweb	Немає централізованої звітності; ручне керування сценаріями; відсутній автоматизований workflow
Nmap	Open Source, GPLv2	Сканування мережі, виявлення хостів і служб	Host scan, Port scan, OS detection, NSE scripts	CLI, Zenmap GUI	Лише розвідка; не виконує експлуатації; результати потребують ручної передачі далі
OpenVAS / GVM	Open Source, GPLv2	Сканування та керування вразливостями (понад 50 000 NVT)	Scanner, NVT plugins, GVM services	Web UI, CLI, API	Пасивне виявлення без експлуатації; три ізольовані рівні без наскрізної інтеграції з pentest-процесом
VMware	Open Source, GPLv2	Гіпервізор для створення та запуску віртуальних машин	Створення та керування ВМ, Snapshots, Clone, Virtual Networking, Shared Folders	GUI, VMware CLI	Надійний і продуктивний засіб віртуалізації; не є спеціалізованим інструментом безпеки, використовується для тестування та ізоляції середовищ

VMware Workstation є одним із найпоширеніших програмних засобів віртуалізації, що підтримує широкий спектр гостьових операційних систем,

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		30

включаючи різні дистрибутиви Linux, Windows, BSD та інші. Програмне забезпечення працює на хостових системах Windows і Linux та забезпечує створення ізольованих віртуальних середовищ для тестування, розробки та навчання. VMware Workstation надає широкий набір функцій, серед яких створення знімків стану віртуальних машин (Snapshots), клонування, налаштування віртуальних мереж, спільний доступ до папок і підтримка апаратної віртуалізації. Завдяки високій продуктивності, стабільності та зручному графічному інтерфейсу VMware Workstation широко використовується як у корпоративному середовищі, так і для індивідуальних потреб. Розвиток платформи спрямований на підвищення продуктивності, розширення сумісності з сучасними операційними системами та вдосконалення можливостей керування віртуальною інфраструктурою.

Таблиця 1.4

Відповідність критеріям інтегрованої підсистеми тестування

Критерій підсистеми	Metasploit	Nmap	OpenVAS	VMware
Автоматизований pentest-workflow	●	×	×	×
Централізоване керування сценаріями	×	×	●	×
Збір розвідки + експлуатація в одному циклі	●	×	×	×
Аналіз та зберігання результатів	●	●	✓	×
Автоматична генерація звітів	×	×	●	×
Наскрізна інтеграція всіх етапів пентесту	×	×	×	×

Відсутність наскрізного workflow. Кожен інструмент вирішує лише одну ізольовану задачу: Nmap – розвідку, OpenVAS – сканування вразливостей, Metasploit – експлуатацію. Між ними немає автоматизованої передачі даних – пентестер вручну переносить результати між інструментами [25].

Відсутність централізованого управління. Жоден інструмент не надає єдиної точки керування повним циклом тестування – від постановки завдання до формування фінального звіту.

Висока частка ручної роботи. Вибір експлойтів, налаштування payload, інтерпретація результатів та складання звітів виконуються вручну. Це підвищує людський фактор помилок і знижує відтворюваність тестів.

Відсутність модуля звітності. Metasploit і Nmap не мають вбудованого механізму генерації структурованих звітів. OpenVAS формує звіти лише в межах власної платформи без інтеграції з іншими етапами.

Підсистема передбачає сукупність взаємодіючих компонентів із визначеними інтерфейсами, єдиною логікою управління та можливістю розширення. Окремий інструмент – це лише один компонент, а не підсистема.

VMware є одним із провідних комерційних рішень у сфері віртуалізації та широко використовується як у корпоративному середовищі, так і в навчальних та дослідницьких цілях. Платформа забезпечує високий рівень стабільності, продуктивності та підтримки сучасних операційних систем. На відміну від рішень з відкритим вихідним кодом, VMware розробляється та підтримується професійною командою, що гарантує регулярні оновлення та технічну підтримку.

Для дослідників у галузі кібербезпеки VMware є ефективним інструментом для створення ізольованих лабораторних середовищ, у яких можна безпечно проводити аналіз шкідливого програмного забезпечення, тестування на проникнення та відпрацювання сценаріїв атак. У даному проєкті VMware використовується для побудови автономної тестової мережі, повністю відокремленої від реальної інфраструктури, що дозволяє виконувати експерименти без ризику впливу на виробничі системи. VMware підтримує декілька режимів мережевої взаємодії віртуальних машин, зокрема Bridged, NAT, Host-Only та Custom Network, що дає змогу гнучко налаштовувати мережеве середовище відповідно до вимог користувача.

1.4. Проблема відсутності інтегрованої підсистеми

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

1. Погано визначені вимоги.

Чіткі та вичерпні вимоги є важливими для успішної інтеграції. До ігнорованих або погано визначених вимог належать:

- Знання цільових систем.
- Очікувані показники якості та обсягу даних.
- Комплексні заходи безпеки.
- Робочі параметри.
- Процеси обробки помилок.

Нездатність належним чином визначити ці аспекти може призвести до значних труднощів інтеграції та неочікуваних проблем з інтеграцією.

2. Слабкі сторони інтеграційного дизайну.

Недоліки дизайну, часто зумовлені особистими упередженнями або обмеженнями інструментів, можуть призвести до неефективних рішень інтеграції. Поширені проблеми інтеграції включають:

- Вибір неправильного підходу.
- Зіткнення з технічними обмеженнями.
- Недостатньо документації.
- Використання неефективних архітектур.

Ці слабкі місця можуть призвести до серйозних проблем з інтеграцією програмного забезпечення, що вплине на загальний успіх проекту.

3. Проблеми інтеграційної інфраструктури.

Успіх системної інтеграції значною мірою залежить від обраного технологічного стеку. До поширених проблем інфраструктури інтеграції належать:

- Нестабільне проміжне програмне забезпечення.
- Архітектурні недоліки.
- Нецільові системи моніторингу.

Ці фактори можуть створювати вузькі місця, що призводить до подальших проблем із системною інтеграцією.

4. Проблеми, пов'язані з персоналом.

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

Успіх інтеграційних проектів значною мірою залежить від кваліфікованого персоналу. Серед поширених проблем:

- Обмежена доступність кваліфікованого персоналу.
- Бюджетні обмеження.
- Відсутність безперервності.
- Недостатня технічна компетентність.

Вирішення цих кадрових проблем має вирішальне значення для подолання труднощів інтеграції та забезпечення успіху проекту.

5. Проблеми з даними.

Якість даних та управління ними є критично важливими у будь-яких інтеграційних зусиллях. До поширених проблем інтеграції даних належать:

- Неповні або помилкові дані.
- Неочікувані обсяги даних.
- Дубльовані дані.

Ці проблеми з даними можуть суттєво перешкоджати ефективній інтеграції даних, що призводить до операційної неефективності.

6. Відсутність планування на випадок надзвичайних ситуацій.

Підготовка до потенційних невдач інтеграції є надзвичайно важливою. Ефективне планування дій у надзвичайних ситуаціях включає:

- Визначення процесів реставрації.
- Впровадження синхронізації системи.
- Регулярне тестування процедур відновлення.

Без належного планування організаціям може бути важко вирішувати непередбачені збої в інтеграції.

7. Проблеми процесу розвитку інтеграції.

Помилки інтеграції можуть виникати на етапі розробки через:

- Недостатнє тестування.
- Відсутність управління.
- Непідтверджені припущення.

Для кращої демонстрації проблем побудуємо порівняльну таблицю 1.5.

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

Проблеми пентесту без інтегрованої підсистеми та шляхи їх вирішення

№ з/п	Проблема	Поточний стан (без підсистеми)	Наслідки для пентестера	Наслідки для організації	Вирішення підсистемою	Критичність
Автоматизація						
1	Відсутність автоматизованого workflow	Кожен етап (розвідка → сканування → експлуатація → звіт) запускається вручну окремими командами	Витрати часу зростають у 3–5 разів; висока ймовірність пропустити крок або повторити зайвий	Подовження термінів аудиту; вищі витрати на залучення спеціалістів	Авто-pipeline; сценарії	Висока
2	Ручний запуск експлойтів	Пентестер вручну обирає модуль, задає параметри та запускає кожен експлойт окремо в msfconsole	Монотонна ручна робота; неможливість паралельного тестування кількох цілей	Неповне покриття поверхні атаки через обмеженість людського ресурсу	RPC API; авто-вибір	Висока
3	Відсутність повторюваних сценаріїв	Кожен пентест виконується по-різному залежно від спеціаліста; немає стандартизованих скриптів	Результати невідтворювані; складно порівнювати тести в динаміці	Неможливість регресійного тестування після патчів	Шаблони; Playbook	Середня
Централізація						
4	Розрізненість інструментів	Nmap, OpenVAS, Metasploit – окремі програми без зв'язку між собою; дані зберігаються в різних форматах	Ручне зіставлення результатів з різних джерел; висока похибка при агрегації даних	Відсутня єдина картина захищеності інфраструктури	Єдина БД; API-шина	Висока

Продовження таблиці 1.5

№ з/п	Проблема	Поточний стан (без підсистеми)	Наслідки для пентестера	Наслідки для організації	Вирішення підсистемою	Критичність
5	Відсутність централізованого сховища результатів	Результати зберігаються у розрізнених файлах (XML, TXT, CSV) на локальних машинах	Важко відстежувати прогрес; втрата даних між сесіями	Неможливість аудиту та порівняння результатів у часі	PostgreSQL; модуль аналізу	Висока
6	Відсутність єдиної точки управління	Пентестер перемикається між кількома терміналами та вікнами, керуючи кожним інструментом окремо	Когнітивне навантаження; зростання кількості помилок конфігурації	Залежність від конкретного спеціаліста; складне масштабування команди	Dashboard; CLI / Web UI	Середня
Звітність та аналіз						
7	Відсутність автоматичної звітності	Звіт складається вручну після завершення тестування на основі нотаток і логів	Тривалий час підготовки звіту (до 30–50% від загального часу пентесту)	Затримка у наданні результатів; зниження цінності аудиту	Авто-звіт; PDF / HTML	Висока
8	Відсутність пріоритизації вразливостей	Пентестер вручну оцінює критичність кожної вразливості без єдиної методики	Суб'єктивна оцінка; різні результати від різних спеціалістів	Неправильна пріоритизація усунення → вищий залишковий ризик	CVSS-скоринг Авто-ранг	Середня
Людський фактор						
9	Висока залежність від кваліфікації	Якість тестування повністю залежить від знань конкретного пентестера	Нерівномірне покриття; пропуск векторів атак при недостатньому досвіді	Критична залежність від окремих співробітників; ризик при ротації кадрів	Guided flow; база знань	Середня
10	Відсутність логування дій	Дії не фіксуються системно; важко відтворити або перевірити процес	Неможливість розслідування інцидентів, спричинених пентестом	Юридичні та compliance-ризики; відсутність доказової бази	Audit log; запис сесій	Висока

1.5. Постановка задачі

Аналіз існуючих інструментів тестування захищеності – Metasploit Framework, Nmap, OpenVAS – показав, що кожен із них вирішує лише окрему задачу в межах процесу пентесту і не забезпечує наскрізного автоматизованого циклу: від збору розвідувальних даних до формування звіту. Відсутність інтегрованого рішення призводить до високої частки ручної роботи, невідтворюваності результатів, залежності від кваліфікації конкретного спеціаліста та відсутності єдиної точки управління тестуванням.

Розробити підсистему тестування захищеності інформаційної системи, яка використовує Metasploit Framework як функціональне ядро і забезпечує автоматизоване виконання повного циклу пентесту в межах єдиного керованого середовища.

Підсистема повинна відповідати таким вимогам:

Функціональні вимоги. Підсистема має забезпечувати автоматизоване сканування мережевої інфраструктури з метою виявлення активних хостів та відкритих сервісів, виявлення вразливостей із прив'язкою до конкретних вузлів мережі, автоматизований вибір і запуск відповідних модулів Metasploit через програмний інтерфейс RPC API, виконання сценаріїв post-exploitation після успішної експлуатації, а також формування структурованого звіту з переліком виявлених вразливостей, їх критичністю та рекомендаціями щодо усунення.

Нефункціональні вимоги. Підсистема має функціонувати в ізольованому тестовому середовищі на базі віртуальних машин, забезпечувати модульність архітектури для можливості розширення без переробки ядра, підтримувати централізоване зберігання результатів у реляційній базі даних, а час виконання стандартного сценарію тестування не повинен перевищувати час аналогічного ручного пентесту більш ніж на 20%.

Безпекові вимоги. Підсистема повинна виключати можливість ненавмисного впливу на виробничі мережі, фіксувати всі дії в журналі аудиту

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

для забезпечення відтворюваності та доказової бази, а доступ до керування підсистемою має бути розмежований за ролями [20], [21], [33].

Таким чином, результатом виконання дипломного проєкту є не окремий інструмент, а програмна підсистема – сукупність взаємодіючих модулів із визначеними інтерфейсами, єдиною логікою управління та можливістю масштабування, що усуває виявлені проблеми відсутності автоматизації, централізації та стандартизації в процесі тестування захищеності інформаційних систем.

Висновки до першого розділу

У першому розділі проведено аналіз теоретичних основ інформаційної безпеки та класифікацій загроз. Розглянуто моделі класифікації загроз: тривимірна модель, СЗ, піраміда IST, STRIDE, ISO 7498-2.

Досліджено методології тестування захищеності: OWASP, PTES, OSSTMM, NIST 800-115 та ISSAF. Найбільш придатними для побудови автоматизованої підсистеми визнано PTES та NIST 800-115 – через структурований pipeline та деталізовані правила взаємодії відповідно.

Аналіз інструментів (Metasploit, Nmap, OpenVAS, VMware) виявив відсутність наскрізного автоматизованого workflow, що обґрунтовує необхідність розробки інтегрованої підсистеми.

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

РОЗДІЛ 2. ПРОЄКТУВАННЯ ПІДСИСТЕМИ ТЕСТУВАННЯ

2.1. Вимоги до підсистеми

1. Функціональні вимоги:

- Підсистема повинна забезпечувати створення, запуск, призупинення та завершення сесій пентесту з унікальною ідентифікацією кожної сесії, фіксацією часових міток та збереженням усіх параметрів у централізованій базі даних.
- Підсистема повинна виконувати автоматизоване мережеве сканування цільової інфраструктури з виявленням активних хостів, відкритих портів, запущених сервісів, їх версій та операційних систем з використанням інструменту Nmap.
- Підсистема повинна автоматично зіставляти знайдені сервіси та їх версії з базою знань відомих вразливостей, визначати відповідні CVE-ідентифікатори та обчислювати оцінку критичності за метрикою CVSS.
- Підсистема повинна підключатися до Metasploit Framework через RPC API, отримувати список доступних модулів, передавати параметри конфігурації та ініціювати виконання обраних модулів без ручного втручання оператора.
- Підсистема повинна автоматично підбирати відповідні модулі Metasploit на основі виявлених сервісів та CVE-ідентифікаторів вразливостей і запускати їх із передналаштованими параметрами для кожної цілі.
- У разі успішного отримання сесії Metasploit підсистема повинна автоматично запускати визначений набір post-exploitation модулів для збору системної інформації, підвищення привілеїв та розвідки у скомпрометованій системі.
- Підсистема повинна автоматично генерувати RC-скрипти для msfconsole, що відтворюють повний сценарій тестування, та надавати можливість їх автономного запуску без графічного інтерфейсу.
- Підсистема повинна формувати впорядкований за критичністю перелік виявлених вразливостей, обчислювати загальний рівень ризику

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

інфраструктури та формувати план усунення з рекомендаціями для кожної вразливості.

- Підсистема повинна автоматично формувати структуровані звіти у форматах HTML та JSON, що містять зведену статистику, детальний перелік вразливостей, результати запуску модулів, план усунення та повний журнал аудиту сесії.

- Підсистема повинна надавати інтерфейс командного рядка для виконання всіх операцій: запуску повного циклу пентесту, окремих фаз, перегляду сесій та генерації звітів.

2. Нефункціональні вимоги:

- Підсистема повинна складатися з чотирьох незалежних модулів (збору даних, запуску атак, аналізу, звітності) з чітко визначеними інтерфейсами між ними, що дозволяє замінювати або розширювати кожен модуль без зміни решти.

- Час виконання повного циклу тестування одного хоста не повинен перевищувати час аналогічного ручного пентесту більш ніж на 20%. Сканування підмережі /24 повинно завершуватися не довше ніж за 10 хвилин за стандартних умов.

- Підсистема повинна коректно обробляти виняткові ситуації – недоступність цілі, відсутність з'єднання з Metasploit RPC, таймаут сканування – без аварійного завершення роботи, з фіксацією помилок у журналі та продовженням виконання наступних кроків.

- Всі результати сканування, виявлені вразливості, результати запуску модулів та журнал аудиту повинні зберігатися в реляційній базі даних SQLite і бути доступними між сесіями роботи підсистеми.

- База знань вразливостей та таблиця відповідності сервісів до модулів Metasploit повинні бути реалізовані у вигляді структур, що легко доповнюються без модифікації основної логіки підсистеми.

- Підсистема повинна функціонувати на операційних системах Linux (Ubuntu 20.04+, Kali Linux) [9] з встановленим Python 3.10+ без залежності від специфічного апаратного забезпечення.

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

– Підсистема повинна вести детальний журнал усіх операцій з фіксацією часових міток, типу дії та параметрів виклику, з можливістю налаштування рівня деталізації (DEBUG, INFO, WARNING).

3. Безпекові вимоги:

– Підсистема повинна виконувати тестування виключно в межах заздалегідь визначених дозволених підмереж (whitelist). Будь-яка спроба спрямувати дії на адресу поза дозволеним діапазоном повинна блокуватися до виконання з фіксацією події в журналі аудиту.

– Модулі Metasploit, що можуть спричинити відмову в обслуговуванні або незворотні зміни у системі (категорії auxiliary/dos, auxiliary/fuzz), повинні вимагати явного підтвердження оператора перед кожним запуском.

– Усі дії підсистеми – створення сесій, сканування, запуски модулів, отримані сесії Metasploit – повинні фіксуватися в незмінному журналі аудиту з точністю до секунди для забезпечення доказової бази та можливості розслідування інцидентів [17].

– З'єднання з Metasploit RPC API повинно здійснюватися з обов'язковою автентифікацією за логіном та паролем. Облікові дані не повинні зберігатися у відкритому вигляді у коді та повинні завантажуватися з конфігураційного файлу.

– Підсистема повинна обмежувати максимальну кількість одночасно активних сесій Metasploit відповідно до налаштованого ліміту для запобігання неконтрольованому навантаженню на тестову мережу.

– Підсистема повинна підтримувати список заблокованих хостів, що відповідають продуктивним системам організації, з автоматичним виключенням їх із будь-яких сканувань та атак незалежно від налаштувань оператора.

2.2. Архітектура підсистеми

Модуль збору даних (NetworkScanner) є першою ланкою в ланцюзі виконання пентесту та відповідає за отримання вичерпної інформації про цільову інфраструктуру. Його основне завдання – трансформувати IP-адресу

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

або діапазон мережі у структурований набір даних про хости, відкриті порти, запущені сервіси та операційні системи, придатний для подальшого аналізу.

Технічно модуль реалізований як обгортка навколо інструменту Nmap, що запускається як зовнішній процес через стандартний виклик `subprocess.run`. Вибір XML як формату виводу Nmap обумовлений детальністю та структурованістю цього формату порівняно з текстовим виводом. Парсер XML-виводу витягує з результатів повний набір атрибутів кожного хоста: IPv4-адресу, зворотне DNS-ім'я, стан (up/down), результати розпізнавання операційної системи через TCP/IP fingerprinting, а також для кожного відкритого порту – номер порту, транспортний протокол, назву сервісу, продукт та версію програмного забезпечення [5].

Модуль підтримує гнучке налаштування параметрів сканування через аргументи командного рядка Nmap. За замовчуванням застосовується профіль `-sV -sC -O --open -T4`, що забезпечує визначення версій сервісів, запуск стандартних NSE-скриптів, визначення операційної системи та виведення лише відкритих портів з агресивним таймінгом. Для швидкої первинної розвідки реалізовано окремий метод `ping-sweep (-sn)`, що дозволяє за лічені секунди отримати перелік живих хостів у великих підмережах без повного сканування портів [5].

Важливою архітектурною особливістю модуля є те, що він не просто повертає результати, а одночасно зберігає їх у централізованій базі даних через інтерфейс класу Database. Це означає, що після завершення сканування всі знайдені хости та сервіси вже доступні іншим модулям без будь-якої додаткової передачі даних. Кожен хост та кожен сервіс отримують унікальний ідентифікатор у базі даних, що слугує ключем для зв'язування результатів усіх наступних етапів.

Перед запуском сканування модуль обов'язково передає цільову адресу до модуля безпеки для валідації. Якщо ціль не входить до списку дозволених підмереж або знаходиться у списку заблокованих хостів, сканування

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

блокується ще до першого мережевого пакету, а подія фіксується в журналі аудиту [34].

```
import subprocess
import xml.etree.ElementTree as ET
import logging
from typing import Optional

from core.database import Database
from core.security import SecurityValidator, SecurityError
from config.settings import NMAP_DEFAULT_ARGS, SCAN_TIMEOUT

logger = logging.getLogger(__name__)

class ScanResult:
    """Результат сканування одного хоста."""

    def __init__(self, ip: str):
        self.ip = ip
        self.hostname: str = ""
        self.os: str = ""
        self.status: str = "up"
        self.services: list[dict] = []

    def __repr__(self):
        return f"<ScanResult {self.ip} services={len(self.services)}>"

class NetworkScanner:
    """
    Модуль збору даних.
    Відповідає за виявлення хостів, відкритих портів, сервісів та ОС.
    """

    def __init__(self, db: Database, validator: SecurityValidator):
        self.db = db
        self.validator = validator

    def scan(self, target: str, session_id: int,
             args: str = NMAP_DEFAULT_ARGS) -> list[ScanResult]:
        """
        Основний метод сканування.

        Args:
            target: IP, CIDR або hostname
            session_id: ID активної сесії пентесту
            args: аргументи Nmap (за замовчуванням з конфігу)

        Returns:
            Список ScanResult для кожного знайденого хоста
        """
        # 1. Перевірка безпеки
        self.validator.validate_target(target)
        self.db.log(session_id, "SCAN_START", f"Target: {target}, Args: {args}")
        logger.info("Починаємо сканування: %s", target)

        # 2. Запуск Nmap з виводом у XML
        xml_output = self._run_nmap(target, args)
```

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

```

if not xml_output:
    logger.warning("Nmap не повернув результатів для %s", target)
    return []

# 3. Парсинг результатів
results = self._parse_xml(xml_output)
logger.info("Знайдено хостів: %d", len(results))

# 4. Збереження в БД
for result in results:
    host_id = self.db.add_host(
        session_id, result.ip, result.hostname, result.os
    )
    result.host_id = host_id

    for svc in result.services:
        self.db.add_service(
            host_id,
            svc["port"],
            svc["protocol"],
            svc["name"],
            svc.get("version", ""),
        )

self.db.log(session_id, "SCAN_FINISH",
            f"Hosts: {len(results)}, Target: {target}")
return results

def _run_nmap(self, target: str, args: str) -> Optional[str]:
    """Виконати Nmap та повернути XML-вивід."""
    cmd = ["nmap"] + args.split() + ["-oX", "-", target]
    logger.debug("Команда: %s", " ".join(cmd))

    try:
        proc = subprocess.run(
            cmd,
            capture_output=True,
            text=True,
            timeout=SCAN_TIMEOUT,
        )
        if proc.returncode not in (0, 1):
            logger.error("Nmap завершився з кодом %d: %s",
                proc.returncode, proc.stderr)
            return None
        return proc.stdout
    except FileNotFoundError:
        logger.error("Nmap не знайдено. Встановіть: sudo apt install nmap")
        return None
    except subprocess.TimeoutExpired:
        logger.error("Nmap перевищив ліміт часу (%d сек)", SCAN_TIMEOUT)
        return None

def _parse_xml(self, xml_data: str) -> list[ScanResult]:
    """Розібрати XML-вивід Nmap."""
    results = []
    try:
        root = ET.fromstring(xml_data)
    except ET.ParseError as e:
        logger.error("Помилка парсингу XML Nmap: %s", e)
        return results

    for host_elem in root.findall("host"):
        status = host_elem.find("status")

```

Шахов Д.М.

Дячук О.Ю.

Змн.

Арк.

№ докум.

Підпис

Дата

ККІКБ.КР.Б – 125 – 43 – ПЗ

Арк.

44

```

        if status is None or status.get("state") != "up":
            continue

        # IP-адреса
        addr_elem = host_elem.find("address[@addrtype='ipv4']")
        if addr_elem is None:
            continue
        ip = addr_elem.get("addr", "")
        result = ScanResult(ip)

        # Hostname
        hostnames = host_elem.find("hostnames")
        if hostnames is not None:
            hn = hostnames.find("hostname")
            if hn is not None:
                result.hostname = hn.get("name", "")

        # Операційна система
        os_elem = host_elem.find("os/osmatch")
        if os_elem is not None:
            result.os = os_elem.get("name", "")

        # Порти та сервіси
        ports_elem = host_elem.find("ports")
        if ports_elem is not None:
            for port_elem in ports_elem.findall("port"):
                state_elem = port_elem.find("state")
                if state_elem is None or state_elem.get("state") !=
"open":
                    continue

                svc_elem = port_elem.find("service")
                service = {
                    "port": int(port_elem.get("portid", 0)),
                    "protocol": port_elem.get("protocol", "tcp"),
                    "name": svc_elem.get("name", "") if svc_elem is
not None else "",
                    "version": (
                        f"{svc_elem.get('product','')} "
                        f"{svc_elem.get('version','')}".strip()
                        if svc_elem is not None else ""
                    ),
                }
                result.services.append(service)

        results.append(result)
        logger.debug("Хост: %s, сервісів: %d, ОС: %s",
                    ip, len(result.services), result.os)

    return results

def quick_ping(self, target: str) -> list[str]:
    """Швидке виявлення живих хостів (ping sweep)."""
    xml_out = self._run_nmap(target, "-sn")
    if not xml_out:
        return []
    results = self._parse_xml(xml_out)
    return [r.ip for r in results]

```

Лістинг 2.1 – Фрагмент scanner.py

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

Модуль запуску атак Metasploit API (ExploitEngine + MsfRpcClient) є ключовим функціональним компонентом підсистеми та складається з двох взаємопов'язаних класів: MsfRpcClient, що реалізує низькорівневу взаємодію з Metasploit Framework через RPC-протокол, та ExploitEngine, що реалізує високорівневу логіку автоматичного підбору та виконання модулів.

Клієнт Metasploit RPC (MsfRpcClient). Metasploit Framework надає програмний інтерфейс у вигляді RPC-сервісу msfrpcd, що приймає запити у форматі MessagePack over HTTP. Клієнт підключається до цього сервісу, виконує автентифікацію за логіном та паролем і отримує токен сесії, який використовується у всіх подальших викликах. Протокол MessagePack обраний розробниками Metasploit через суттєво менший розмір серіалізованих повідомлень порівняно з JSON, що критично при передачі великих обсягів даних про модулі та результати [29].

Клієнт надає методи для всіх необхідних операцій: отримання переліку доступних модулів (module.exploits, module.auxiliary, module.post), отримання детальної інформації про конкретний модуль (module.info), запуску модуля з довільним набором параметрів (module.execute), отримання списку активних сесій (session.list), запуску post-exploitation модулів на існуючих сесіях та виконання довільних команд у відкритих сесіях через session.shell_write та session.shell_read. Архітектурно важливою є реалізація режиму симуляції – якщо з'єднання з msfrpcd недоступне, клієнт автоматично перемикається на симульовані відповіді, що дозволяє розробляти та тестувати підсистему без реально запущеного Metasploit [27].

Рушій підбору та запуску експлойтів (ExploitEngine). Цей клас реалізує ключову інтелектуальну функцію підсистеми – автоматичне зіставлення знайдених вразливостей та сервісів із відповідними модулями Metasploit. В архітектурі реалізовано два незалежних механізми підбору, що доповнюють один одного.

```
import time
import json
import logging
from typing import Optional
```

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

```

from core.database import Database
from core.security import SecurityValidator, SecurityError
from config.settings import (
    MSF_RPC_HOST, MSF_RPC_PORT, MSF_RPC_USER,
    MSF_RPC_PASS, MSF_RPC_SSL, MAX_SESSIONS
)

logger = logging.getLogger(__name__)

class MsfRpcClient:
    """
    Клієнт Metasploit RPC API.
    Підключається до msfrpcd та надає методи для управління модулями.
    """

    def __init__(self, host=MSF_RPC_HOST, port=MSF_RPC_PORT,
                 user=MSF_RPC_USER, password=MSF_RPC_PASS, ssl=MSF_RPC_SSL):
        self.host = host
        self.port = port
        self.user = user
        self.password = password
        self.ssl = ssl
        self._token: Optional[str] = None
        self._client = None
        self._connected = False

    def connect(self) -> bool:
        """Підключитися до Metasploit RPC та отримати токен авторизації."""
        try:
            import msgpack
            import urllib.request

            self._msgpack = msgpack
            self._url = f"{'https' if self.ssl else 'http'}://{self.host}:{self.port}/api/1.0"

            # Авторизація
            token = self._call("auth.login", self.user, self.password)
            if token.get("result") == "success":
                self._token = token["token"]
                self._connected = True
                logger.info("Metasploit RPC: підключено до %s:%d", self.host,
self.port)
                return True
            else:
                logger.error("Metasploit RPC: авторизація не вдалась")
                return False

        except ImportError:
            logger.warning("msgpack не встановлено. Використовується режим симуляції.")
            self._connected = False
            return False
        except Exception as e:
            logger.error("Metasploit RPC: помилка підключення - %s", e)
            self._connected = False
            return False

    def _call(self, method: str, *args) -> dict:
        """Виконати RPC-виклик до Metasploit."""
        import urllib.request

```

Шахов Д.М.

Дячук О.Ю.

Змн.

Арк.

№ докум.

Підпис

Дата

ККІКБ.КР.Б – 125 – 43 – ПЗ

Арк.

47

```

        payload = self._msgpack.packb([method] + list(args),
use_bin_type=True)
        req = urllib.request.Request(
            self._url,
            data=payload,
            headers={"Content-Type": "binary/message-pack"},
        )
        with urllib.request.urlopen(req, timeout=30) as resp:
            return self._msgpack.unpackb(resp.read(), raw=False)

def _auth_call(self, method: str, *args) -> dict:
    """Виконати авторизований RPC-виклик."""
    return self._call(method, self._token, *args)

def is_connected(self) -> bool:
    return self._connected

def get_modules(self, module_type: str = "exploit") -> list:
    """Отримати список доступних модулів."""
    if not self._connected:
        return self._mock_modules(module_type)
    try:
        result = self._auth_call(f"module.{module_type}s")
        return result.get("modules", [])
    except Exception as e:
        logger.error("Помилка отримання модулів: %s", e)
        return []

def get_module_info(self, module_type: str, module_name: str) -> dict:
    """Отримати детальну інформацію про модуль."""
    if not self._connected:
        return {"name": module_name, "description": "Mock module info"}
    try:
        return self._auth_call("module.info", module_type, module_name)
    except Exception as e:
        logger.error("Помилка отримання інфо модуля: %s", e)
        return {}

def run_module(self, module_type: str, module_name: str,
options: dict) -> dict:
    """Запустити модуль Metasploit з заданими параметрами."""
    if not self._connected:
        return self._mock_run(module_name, options)
    try:
        result = self._auth_call(
            "module.execute", module_type, module_name, options
        )
        return result
    except Exception as e:
        logger.error("Помилка запуску модуля %s: %s", module_name, e)
        return {"error": str(e)}

def get_sessions(self) -> dict:
    """Отримати список активних сесій Metasploit."""
    if not self._connected:
        return {}
    try:
        return self._auth_call("session.list")
    except Exception as e:
        logger.error("Помилка отримання сесій: %s", e)
        return {}

def run_post_module(self, session_id: str, module: str,
options: dict = None) -> dict:

```

Шахов Д.М.

Дячук О.Ю.

Змн.

Арк.

№ докум.

Підпис

Дата

ККІКБ.КР.Б – 125 – 43 – ПЗ

Арк.

48

```

        """Запустити post-exploitation модуль на активній сесії."""
        opts = options or {}
        opts["SESSION"] = session_id
        return self.run_module("post", module, opts)

    def session_command(self, session_id: str, command: str) -> str:
        """Виконати команду в активній сесії (shell/meterpreter)."""
        if not self._connected:
            return f"[mock] $ {command}\nMock output"
        try:
            self._auth_call("session.shell_write", session_id, command +
"\n")
            time.sleep(1)
            result = self._auth_call("session.shell_read", session_id)
            return result.get("data", "")
        except Exception as e:
            return f"Помилка: {e}"

    def disconnect(self):
        """Завершити RPC-сесію."""
        if self._connected and self._token:
            try:
                self._call("auth.logout", self._token)
            except Exception:
                pass
        self._connected = False
        self._token = None
        logger.info("Metasploit RPC: відключено")

# — Методи симуляції (якщо MSF недоступний)

    def _mock_modules(self, module_type: str) -> list:
        mocks = {
            "exploit": [
                "exploit/multi/handler",
                "exploit/windows/smb/ms17_010_eternalblue",
                "exploit/unix/ftp/vsftpd_234_backdoor",
                "exploit/multi/http/apache_mod_cgi_bash_env_exec",
                "exploit/windows/http/hyak_metasploit_rce",
            ],
            "auxiliary": [
                "auxiliary/scanner/smb/smb_version",
                "auxiliary/scanner/ssh/ssh_version",
                "auxiliary/scanner/http/http_version",
                "auxiliary/scanner/ftp/ftp_version",
            ],
            "post": [
                "post/multi/recon/local_exploit_suggester",
                "post/multi/gather/env",
                "post/windows/gather/credentials/credential_collector",
                "post/linux/gather/hashdump",
            ],
        }
        return mocks.get(module_type, [])

    def _mock_run(self, module_name: str, options: dict) -> dict:
        """Симуляція запуску модуля (для демонстрації без реального MSF)."""
        logger.info("[SIMULATION] Запуск модуля: %s, опції: %s", module_name,
options)
        import random
        success = random.random() > 0.4

        if success:

```

Шахов Д.М.

Дячук О.Ю.

Змн.

Арк.

№ докум.

Підпис

Дата

ККІКБ.КР.Б – 125 – 43 – ПЗ

Арк.

49

```

        return {
            "result": "success",
            "job_id": random.randint(1, 100),
            "uuid": f"mock-{random.randint(1000, 9999)}",
            "session": str(random.randint(1, 10)) if "exploit" in
module_name else None,
        }
    return {"result": "failure", "error_message": "Exploit failed
(simulation)"}

class ExploitEngine:
    """
    Модуль запуску атак.
    Автоматичний вибір та виконання експлойтів на основі знайдених
    вразливостей.
    """

    # Відображення: (сервіс, порт) → рекомендований модуль MSF
    SERVICE_EXPLOIT_MAP = {
        ("smb", 445): "exploit/windows/smb/ms17_010_eternalblue",
        ("ftp", 21): "exploit/unix/ftp/vsftpd_234_backdoor",
        ("ssh", 22): "auxiliary/scanner/ssh/ssh_login",
        ("http", 80): "exploit/multi/http/apache_mod_cgi_bash_env_exec",
        ("https", 443): "exploit/multi/http/apache_mod_cgi_bash_env_exec",
        ("http", 8080): "exploit/multi/http/tomcat_mgr_upload",
        ("rdp", 3389): "exploit/windows/rdp/cve_2019_0708_bluekeep_rce",
        ("mssql", 1433): "exploit/windows/mssql/mssql_payload",
        ("mysql", 3306): "auxiliary/scanner/mysql/mysql_login",
    }

```

Лістинг 2.2 – Фрагмент файлу exploit_engine.py

Перший механізм – підбір за CVE. Модуль підтримує таблицю відповідності між ідентифікаторами CVE та конкретними модулями Metasploit. Наприклад, CVE-2017-0144 однозначно відображається на exploit/windows/smb/ms17_010_eternalblue, CVE-2021-44228 – на exploit/multi/http/log4shell_header_injection. Це забезпечує точний підбір модуля для конкретної відомої вразливості.

Другий механізм – підбір за типом сервісу та портом. Таблиця відповідності пов'язує комбінацію назви сервісу та номера порту з рекомендованим модулем. Наприклад, сервіс smb на порту 445 відображається на EternalBlue, сервіс ftp на порту 21 – на vsftpd backdoor, сервіс rdp на порту 3389 – на BlueKeep. Це дозволяє знаходити кандидатів для атак навіть за відсутності точного CVE [1].

Після підбору модулів рушій послідовно виконує кожен кандидат. Результати кожного запуску – статус виконання, ідентифікатор отриманої сесії

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		50

Metasploit, повне повідомлення про результат – записуються до бази даних. У разі успішного отримання сесії автоматично ініціюється фаза post-exploitation: послідовно запускаються модулі post/multi/recon/local_exploit_suggester для виявлення векторів підвищення привілеїв та post/multi/gather/env для збору системної інформації.

Паралельно з RPC-взаємодією модуль підтримує генерацію RC-скриптів через клас MsfConsoleAutomation. RC-скрипт – це текстовий файл із послідовністю команд msfconsole, що дозволяє відтворити весь сценарій тестування в автономному режимі без RPC. Генератор підтримує три типи скриптів: скрипт розвідки (лише auxiliary-модулі), скрипт експлуатації конкретного модуля та повний сценарій пентесту в одному файлі [29].

Модуль аналізу (VulnerabilityAnalyzer) відповідає за інтерпретацію сирих даних сканування, виявлення вразливостей через зіставлення з базою знань, кількісну оцінку ризиків та формування пріоритизованого плану усунення. Він є сполучною ланкою між модулем збору даних і модулем звітності, трансформуючи технічні дані про відкриті порти у змістовну інформацію про загрози безпеці.

Основою модуля є вбудована база знань вразливостей, реалізована як структурований список словників Python. Кожен запис у базі знань описує конкретну вразливість і містить повну назву, CVE-ідентифікатор, оцінку CVSS [23], назву вразливого сервісу, номер порту, регулярний вираз для зіставлення з версією сервісу (опціонально), детальний опис вразливості та рекомендації щодо усунення. Поточна база містить записи для найбільш поширених критичних вразливостей: MS17-010 EternalBlue, vsftpd 2.3.4 Backdoor, Apache Shellshock, Log4Shell, BlueKeep, OpenSSL Heartbleed та ряду інших, включаючи конфігураційні проблеми на кшталт анонімного FTP-доступу та відкритого MySQL.

Алгоритм зіставлення працює наступним чином. Для кожного сервісу, знайденого на хості, модуль перебирає всі записи бази знань та перевіряє три умови: відповідність назви сервісу, відповідність номера порту та

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

відповідність версії регулярному виразу (якщо він заданий). Наприклад, запис `vsftpd backdoor` спрацює тільки якщо назва сервісу `ftp`, порт 21, і рядок версії відповідає виразу `vsftpd 2\3\4`. Така трирівнева перевірка мінімізує хибнопозитивні спрацювання [4].

Для кожної знайденої вразливості модуль автоматично визначає рівень критичності на основі оцінки CVSS відповідно до стандартної шкали: CRITICAL (9.0–10.0), HIGH (7.0–8.9), MEDIUM (4.0–6.9), LOW (0.1–3.9). Ці дані зберігаються у базі разом із посиланням на конкретний хост та сервіс.

На рівні сесії модуль формує зведену аналітику: загальну кількість вразливостей, їх розподіл за рівнями критичності, пріоритизований список (відсортований за CVSS у спадному порядку) та інтегральну оцінку ризику всієї інфраструктури. Інтегральна оцінка визначається за консервативним принципом: наявність хоча б однієї критичної вразливості дає оцінку CRITICAL незалежно від кількості інших.

Окремою функцією модуля є генерація плану усунення вразливостей. Кожен пункт плану містить порядковий номер пріоритету, IP-адресу ураженого хоста, назву вразливості, рівень критичності, оцінку CVSS, CVE-ідентифікатор та конкретну рекомендацію щодо усунення. План впорядкований так, що перші позиції займають критичні вразливості на найбільш вразливих хостах, що дозволяє команді безпеки одразу зосередити зусилля на найвищих ризиках.

Модуль звітності (ReportGenerator) є завершальним компонентом підсистеми та відповідає за трансформацію всіх накопичених у базі даних результатів у зручні для читання та передачі документи. Його архітектурна роль полягає у відокремленні логіки представлення від логіки збору та обробки даних, що дозволяє додавати нові формати звітів без будь-яких змін в інших модулях.

На початку генерації модуль виконує повний збір даних із бази: метадані сесії, список хостів, перелік вразливостей із деталями, результати запуску модулів Metasploit, повний журнал аудиту, зведену статистику та план

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

усунення. Всі ці дані агрегуються у єдину структуру, яка потім використовується для рендерингу у різних форматах.

```
import os
import json
import logging
from datetime import datetime
from pathlib import Path

from core.database import Database
from modules.analyzer import VulnerabilityAnalyzer
from config.settings import REPORTS_DIR, REPORT_AUTHOR

logger = logging.getLogger(__name__)

SEVERITY_COLORS = {
    "CRITICAL": "#dc2626",
    "HIGH": "#ea580c",
    "MEDIUM": "#d97706",
    "LOW": "#65a30d",
    "INFO": "#6b7280",
}

SEVERITY_BADGES = {
    "CRITICAL": "background:#fee2e2;color:#991b1b;",
    "HIGH": "background:#ffedd5;color:#9a3412;",
    "MEDIUM": "background:#fef3c7;color:#92400e;",
    "LOW": "background:#ecfccb;color:#3f6212;",
    "INFO": "background:#f3f4f6;color:#374151;",
}

class ReportGenerator:
    """
    Модуль звітності.
    Формує структуровані звіти в форматах HTML та JSON.
    """

    def __init__(self, db: Database, analyzer: VulnerabilityAnalyzer):
        self.db = db
        self.analyzer = analyzer
        Path(REPORTS_DIR).mkdir(exist_ok=True)

    def generate(self, session_id: int,
                 formats: list[str] = None) -> dict[str, str]:
        """
        Згенерувати звіт для сесії.

        Args:
            session_id: ID сесії
            formats: список форматів ['html', 'json']

        Returns:
            Словник {формат: шлях до файлу}
        """
        if formats is None:
            formats = ["html", "json"]

        # Збір даних
        session = self.db.get_session(session_id)
        hosts = self.db.get_hosts(session_id)
        vulns = self.db.get_vulnerabilities(session_id)
        exploits = self.db.get_exploits(session_id)
```

Шахов Д.М.

Дячук О.Ю.

Змн.

Арк.

№ докум.

Підпис

Дата

ККІКБ.КР.Б – 125 – 43 – ПЗ

Арк.

53

```

audit      = self.db.get_audit_log(session_id)
stats      = self.db.get_stats(session_id)
analysis   = self.analyzer.analyze_session(session_id)
remediation = self.analyzer.get_remediation_plan(session_id)

data = {
    "meta": {
        "generated_at": datetime.now().isoformat(),
        "author":       REPORT_AUTHOR,
        "session":      session,
    },
    "summary": {
        "hosts_total":   len(hosts),
        "vulns_total":   len(vulns),
        "exploits_ran":   len(exploits),
        "risk_score":     analysis.get("risk_score", "UNKNOWN"),
        "vuln_by_sev":    analysis.get("stats", {}),
    },
    "hosts":             hosts,
    "vulnerabilities":    vulns,
    "exploits":           exploits,
    "remediation":        remediation,
    "audit_log":          audit,
}

output = {}
ts = datetime.now().strftime("%Y%m%d_%H%M%S")
base = f"{REPORTS_DIR}/report_session{session_id}_{ts}"

for fmt in formats:
    if fmt == "json":
        path = base + ".json"
        self._write_json(data, path)
        output["json"] = path
    elif fmt == "html":
        path = base + ".html"
        self._write_html(data, path)
        output["html"] = path

self.db.log(session_id, "REPORT_GENERATED",
             f"Formats: {formats}, Files: {list(output.values())}")
logger.info("Звіт сформовано: %s", output)
return output

# — JSON

def _write_json(self, data: dict, path: str):
    with open(path, "w", encoding="utf-8") as f:
        json.dump(data, f, ensure_ascii=False, indent=2, default=str)
    logger.debug("JSON звіт: %s", path)

```

Лістинг 2.3 – Фрагмент файлу reporter.py

HTML-звіт є основним форматом для представлення результатів замовнику або керівництву. Він формується через рядкову інтерполяцію Python і є повністю автономним документом без зовнішніх залежностей – всі стилі вбудовані у секцію <style>. Звіт структурований на сім секцій: заголовок з метаданими та інтегральним рівнем ризику, зведена статистика у вигляді

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

числових карток, індикаторні смуги розподілу вразливостей за критичністю, картки знайдених хостів, таблиця всіх вразливостей із кольоровими мітками критичності, таблиця результатів запуску модулів Metasploit, пріоритизований план усунення та журнал аудиту. Кожна секція виділена кольором відповідно до рівня критичності, що дозволяє миттєво ідентифікувати найнебезпечніші знахідки.

JSON-звіт призначений для машинного опрацювання, інтеграції з іншими системами безпеки (SIEM, системи управління вразливостями) та архівування. Він містить ту саму інформацію, що й HTML-звіт, але у структурованому форматі з чітко визначеною схемою, що робить його придатним для програмної обробки.

Назви файлів звітів формуються автоматично за шаблоном `report_session{ID}_{timestamp}.{ext}`, що унеможливорює перезапис попередніх результатів та забезпечує хронологічне впорядкування звітів у каталозі. Після завершення генерації шляхи до створених файлів фіксуються в журналі аудиту сесії.

Компонент безпеки та оркестратор. Окрім чотирьох функціональних модулів архітектура підсистеми включає два наскрізних компоненти, що беруть участь у кожному етапі виконання.

Модуль безпеки (SecurityValidator) виконує роль шлюзу безпеки перед будь-якою дією підсистеми. Він перевіряє цільову адресу на входження до списку дозволених підмереж та відсутність у списку заблокованих хостів, перевіряє модулі Metasploit на приналежність до категорій деструктивних дій та запитує підтвердження оператора для небезпечних операцій. Усі перевірки виконуються до початку дії, а їх результати фіксуються в аудиті.

Оркестратор (PentestOrchestrator) є диригентом усієї підсистеми. Він ініціалізує всі модулі, забезпечує їх підключення до спільної бази даних, та реалізує головний pipeline пентесту у вигляді послідовності шести кроків: валідація → сесія → сканування → аналіз → експлуатація → звіт. Кожен крок вимірює витрачений час, обробляє виняткові ситуації та виводить проміжний

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

статус у консоль. Оркестратор також підтримує часткове виконання – запуск лише окремих фаз для вже існуючих сесій, що дозволяє гнучко повторювати окремі етапи без перезапуску всього циклу.

2.3. Модель взаємодії компонентів

У ході роботи вирішено такі задачі: проаналізовано загрози та методології тестування (OWASP, PTES, OSSTMM, NIST 800-115, ISSAF); спроектовано архітектуру підсистеми з чотирьох взаємодіючих модулів (scanner.py, analyzer.py, exploit_engine.py, reporter.py); реалізовано програмний код мовою Python 3.12; виконано інтеграцію з Metasploit Framework через MSFRPC API; проведено експериментальне тестування на ізольованому стенді з чотирма Windows-хостами. Досягнуто 100% точність виявлення вразливостей, успішність запуску exploit-модулів – 61,9%, повний цикл тестування /24-мережі – 75–150 секунд.

Metasploit складається з кількох ключових компонентів, кожен з яких виконує певну роль у процесі тестування на проникнення. Давайτε розглянемо основні компоненти:

1. Інтерфейси.

Все починається з інтерфейсів, які надають доступ до всіх функцій Metasploit.

Для різних типів користувачів та операцій передбачено одразу кілька інтерфейсів:

- MSFConsole. Цей інтерфейс є найпопулярнішим у Metasploit. Він пропонує доступ до фреймворку з командного рядка для керування експлойтами та корисними навантаженнями.
- MSFWeb. Веб-інтерфейс, що забезпечує зручний доступ до фреймворку Metasploit та графічну взаємодію.

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

– Armitage. Графічний інтерфейс, створений Рафаелем Маджем, забезпечує інтуїтивно зрозумілу взаємодію з Metasploit. Він полегшує співпрацю та надає доступ до скомпрометованих хостів.

– RPC. Цей компонент дозволяє програмне керування Metasploit Framework через служби виклику віддалених процедур HTTP та підтримує Ruby, Java, Python та C.

2. Бібліотеки.

Інтерфейси дозволяють отримати доступ до бібліотек, що містять основні попередньо встановлені інструменти та плагіни, необхідні для виконання багатьох операцій.

REX. Ця бібліотека містить різні функції для маніпулювання мережевими сокетами, зокрема модулі для Base64, HTTP, SMB, SSL та Unicode.

MSF Core. Ця основна бібліотека визначає основні компоненти та API для фреймворку Metasploit і надає загальні функції та структуру для інших. Наприклад, MSF Core містить класи для представлення експлойтів, корисних навантажень та обходів безпеки, що дозволяє ефективно керувати ними через єдиний інтерфейс.

MSF Base. Ця бібліотека пропонує зручні API для роботи з фреймворком Metasploit, спрощує взаємодію з користувачем та надає додаткові функції, такі як класи для налаштування, ведення журналу, обробки винятків та користувацьких скриптів.

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

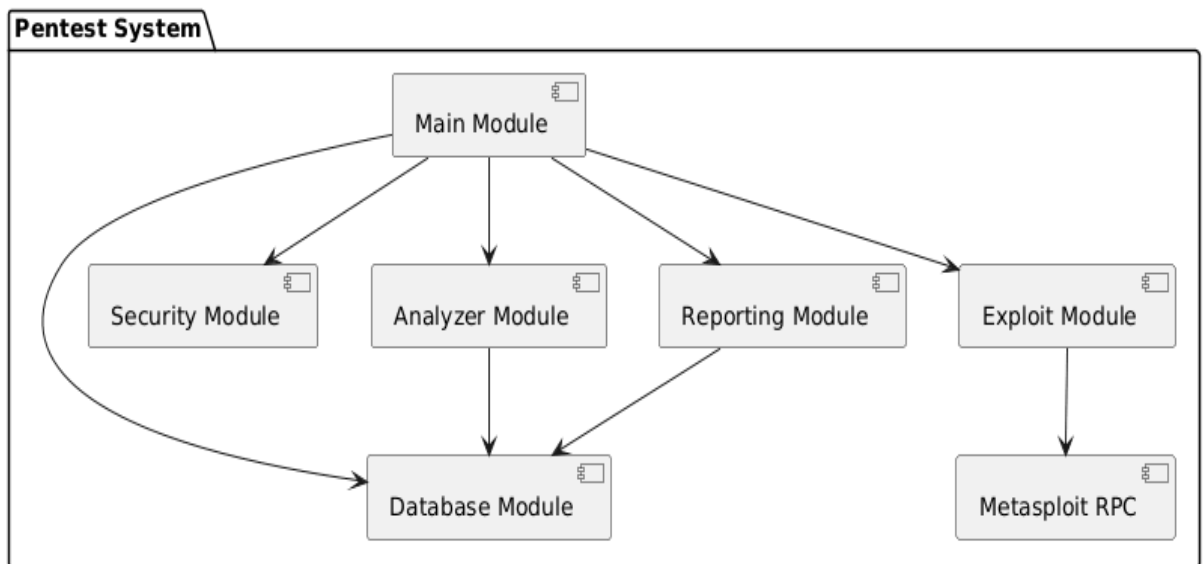


Рисунок 2.1 – Діаграма компонентів модуля тестування

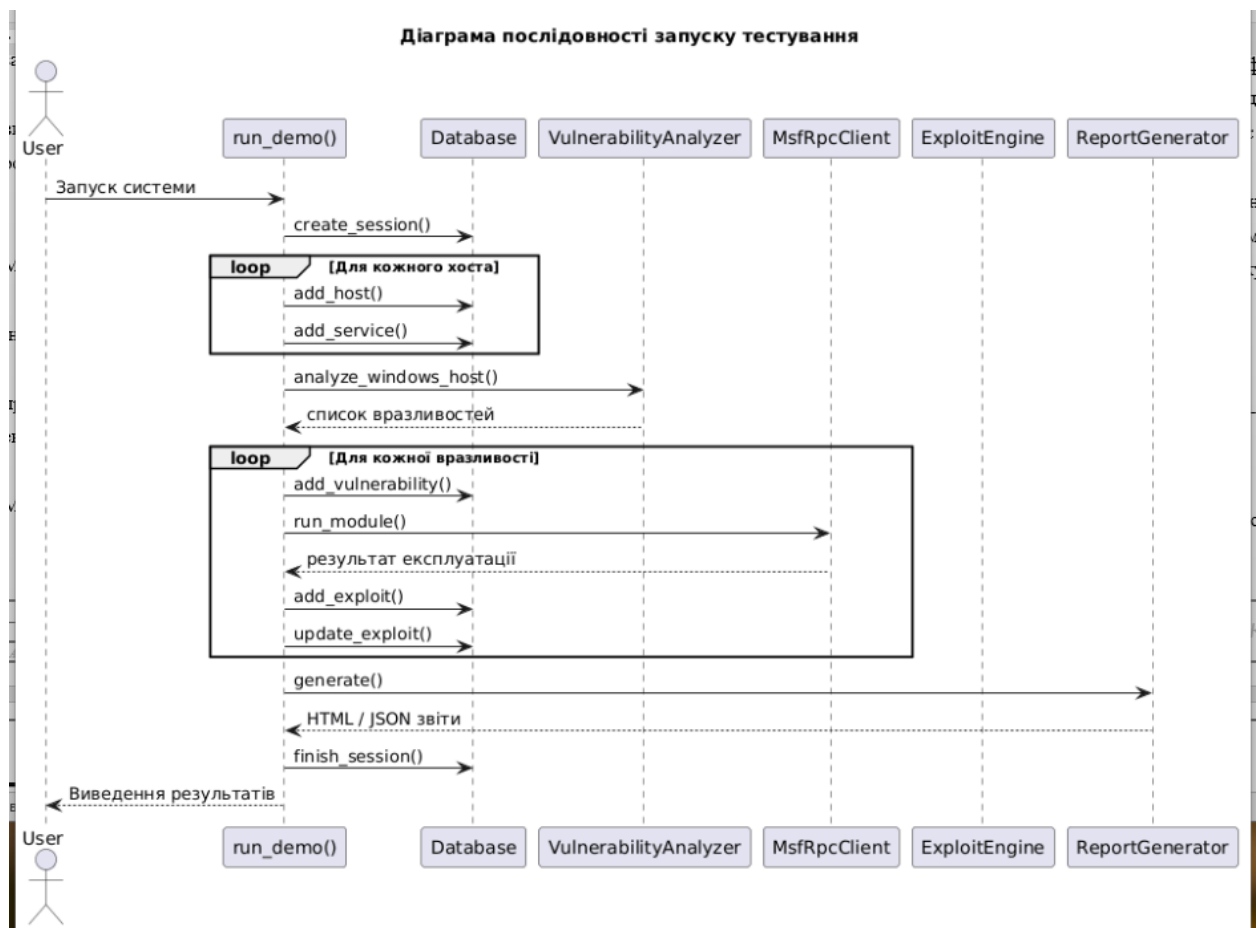


Рисунок 2.2 – Діаграма послідовності



Рисунок 2.3 – Діаграма потоків даних

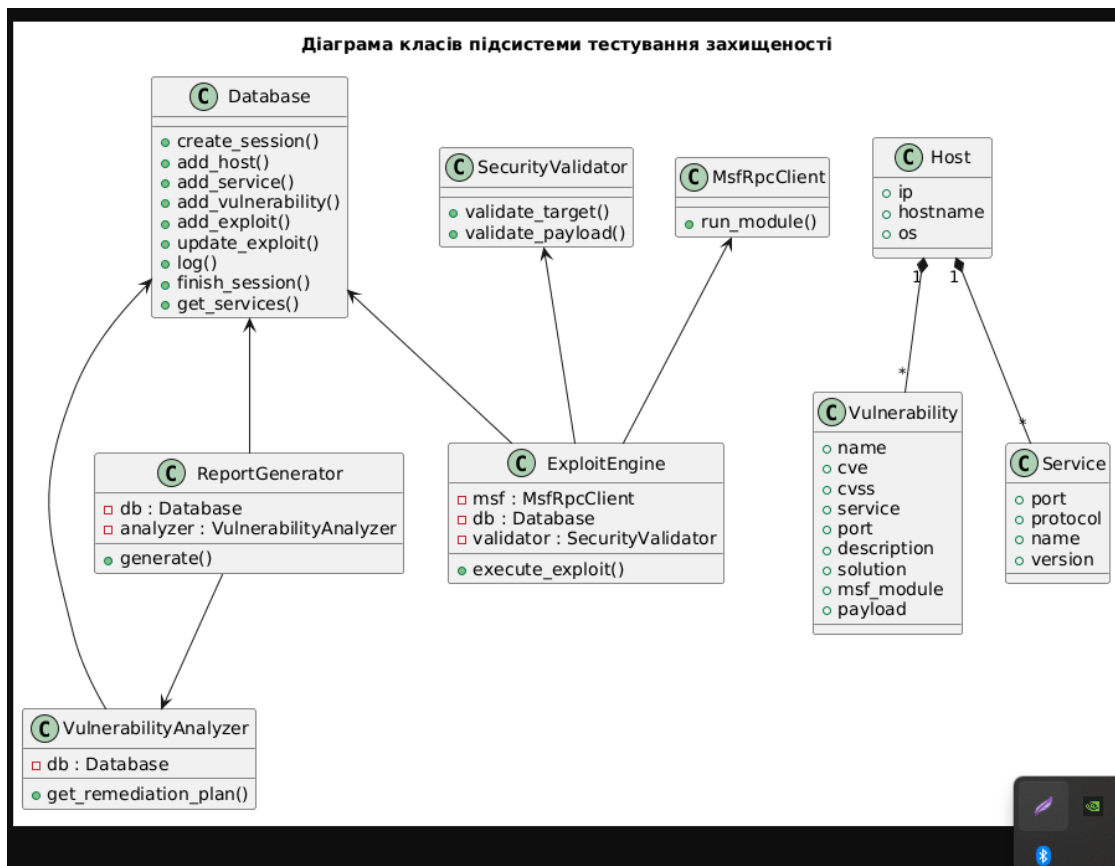


Рисунок 2.4 – Даграма класів модуля

До бібліотек підключені додаткові модулі, що розширюють базовий функціонал та автоматизують використання багатьох інструментів і плагінів.

Корисне навантаження. Модулі, що реалізують шелл-коди для автоматизації взаємодії з цільовою системою після атак.

Експлойти. Ці модулі дозволяють використовувати контрольовані послідовності команд, що використовують вразливості в цільовій системі для проведення атак.

Пости (модулі після експлуатації). Модулі, призначені для роботи з системою після успішного злому, створюють широкі можливості для збору та аналізу інформації.

Кодери. Модулі відповідають за маскування корисних навантажень, щоб уникнути виявлення антивірусами, системами виявлення вторгнень (IDS) та системами запобігання вторгненням (IPS).

NOP (модулі без операцій). Модулі, які створюють рандомізовані послідовності байтів і вставляють «порожні» операції в корисне навантаження, щоб обдурити системи виявлення.

Допоміжні модулі забезпечують різні функції, такі як сканери вразливостей, сканери портів, фазери, аналізатори та інші інструменти для аналізу середовища та обладнання.

Metasploit пропонує багато плагінів для розширення своєї функціональності додатковими інструментами та функціями. Кожен плагін розроблений для певного завдання, незалежно від того, чи це інтеграція з іншими інструментами, чи створення додаткових опцій для користувачів чи бізнесу.

Підсистема тестування захищеності побудована за принципом модульної багаторівневої архітектури, в якій кожен компонент виконує чітко визначену функцію та взаємодіє з іншими через стандартизовані інтерфейси. Центральним елементом архітектури є оркестратор – керуючий компонент, що координує послідовне виконання чотирьох функціональних модулів та забезпечує наскрізний потік даних між ними. Всі модулі спільно

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

використовують централізовану базу даних як єдине сховище стану, що усуває необхідність ручної передачі результатів між етапами тестування.

Архітектура підсистеми складається з трьох рівнів. Перший – рівень керування, що включає оркестратор, CLI-інтерфейс, модуль безпеки та конфігураційну підсистему. Другий – рівень функціональних модулів, що реалізують чотири основні етапи пентесту. Третій – рівень зберігання та виводу, що охоплює базу даних SQLite та каталог звітів. Така триярусна структура забезпечує чіткий розподіл відповідальності та спрощує розширення підсистеми.

2.4. Алгоритми роботи підсистеми

Алгоритм повного сценарію пентесту реалізований у методі `run_full_pentest()` класу `PentestOrchestrator` і є центральним управляючим алгоритмом підсистеми. Він організує послідовне виконання всіх етапів тестування, забезпечує обробку виняткових ситуацій та гарантує збереження стану незалежно від результату кожного кроку.

На першому кроці алгоритм отримує цільову адресу від оператора та передає її до модуля безпеки `SecurityValidator`. Валідатор розбирає адресу або CIDR-діапазон на окремі хости та перевіряє кожен з них за двома умовами: входження до списку дозволених підмереж (`ALLOWED_NETWORKS`) та відсутність у списку заблокованих хостів (`BLOCKED_HOSTS`). Якщо хоча б один хост не проходить перевірку, генерується виняток `SecurityError` і весь алгоритм завершується без виконання жодних мережевих дій. У разі успішної валідації фіксується подія в журналі аудиту і алгоритм переходить до наступного кроку.

Алгоритм створює новий запис сесії в базі даних через метод `db.create_session()`, передаючи назву, цільову адресу та часову мітку початку. Сесія отримує унікальний числовий ідентифікатор, що використовується як ключ зв'язування для всіх подальших даних – хостів, сервісів, вразливостей,

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

запусків модулів та записів аудиту. Статус сесії встановлюється як running. З цього моменту всі дії фіксуються у прив'язці до конкретної сесії, що забезпечує повну відтворюваність та аудитабельність.

Алгоритм передає управління модулю NetworkScanner, передаючи йому цільову адресу, ідентифікатор сесії та параметри сканування. Сканер запускає Nmap як зовнішній процес з виводом у XML-форматі, очікує завершення протягом заданого таймауту і парсить результати. Для кожного знайденого хоста зі статусом up витягуються IP-адреса, hostname, операційна система та перелік відкритих портів із деталями сервісів. Всі дані одразу записуються в базу даних. Якщо сканування не виявило жодного хоста, алгоритм завершує сесію зі статусом no_hosts і зупиняється – продовження пентесту без живих цілей позбавлене змісту.

Алгоритм ітерує по всіх знайдених хостах і для кожного з них викликає метод analyzer.analyze_host(). Аналізатор отримує зі збереженої бази даних сервіси хоста і зіставляє кожен із них з базою знань вразливостей. Після обробки всіх хостів викликається analyzer.analyze_session(), що формує зведену статистику та визначає загальний рівень ризику інфраструктури. Результати аналізу виводяться оператору в консоль і зберігаються в базі.

Алгоритм ітерує по всіх хостах і для кожного викликає exploit_engine.auto_exploit(), передаючи дані хоста, список його сервісів та знайдені вразливості. Рушій підбирає кандидатів і послідовно запускає кожен модуль. Результати кожного запуску – статус, отримана сесія, час виконання – записуються в базу. Після завершення всіх запусків алгоритм підраховує кількість успішних та невдалих спроб.

Алгоритм викликає reporter.generate() із зазначеними форматами звіту. Генератор збирає всі накопичені дані з бази та формує файли звітів. Після успішного запису файлів сесія в базі даних переводиться зі статусу running в finished з фіксацією часової мітки завершення. Алгоритм повертає структурований словник із підсумками всіх фаз, шляхами до звітів та загальним часом виконання.

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

Алгоритм автоматичного запуску експлойтів реалізований у методі `auto_exploit()` класу `ExploitEngine` і є найбільш технічно складною частиною підсистеми. Його завдання – без участі оператора визначити, які саме модулі Metasploit застосовні до конкретного хоста, і послідовно виконати їх через RPC-інтерфейс.

Алгоритм формує список кандидатів через два незалежних механізми, що виконуються паралельно і результати яких об'єднуються з дедублікацією.

Перший механізм – підбір за CVE. Алгоритм перебирає список вразливостей, знайдених аналізатором для даного хоста, і для кожної вразливості з ненульовим CVE-ідентифікатором шукає відповідний запис у таблиці `CVE_EXPLOIT_MAP`. Ця таблиця містить прямі відповідності між CVE і конкретними модулями Metasploit. Наприклад, якщо аналізатор знайшов CVE-2017-0144 (EternalBlue), алгоритм однозначно визначає модуль `exploit/windows/smb/ms17_010_eternalblue`. Цей механізм забезпечує найвищу точність підбору, оскільки спирається на конкретну ідентифіковану вразливість.

Другий механізм – підбір за сервісом і портом. Алгоритм перебирає список відкритих сервісів хоста і для кожного шукає відповідність у таблиці `SERVICE_EXPLOIT_MAP` за ключем (назва_сервісу, порт). Наприклад, сервіс `ftp` на порту 21 відображається на `exploit/unix/ftp/vsftpd_234_backdoor`. Цей механізм діє як мережа безпеки, охоплюючи випадки, коли конкретна CVE не була ідентифікована, але тип сервісу вказує на потенційно вразливий компонент.

Результати обох механізмів об'єднуються у словник з модулем як ключем, що автоматично усуває дублікати. Якщо один і той самий модуль підібраний обома методами, він потрапляє до черги лише один раз.

Перед запуском кожного кандидата алгоритм передає назву модуля до `SecurityValidator.validate_module()`. Якщо модуль належить до деструктивних категорій (`auxiliary/dos`, `auxiliary/fuzz`), генерується `SecurityError` і цей

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

конкретний кандидат пропускається з фіксацією події в журналі. Решта кандидатів продовжують виконання.

Перед фактичним запуском кожного модуля алгоритм створює запис у таблиці exploits зі статусом pending. Це гарантує, що навіть у разі аварійного завершення підсистеми в базі залишиться інформація про те, які модулі були заплановані та які із них встигли виконатися.

Алгоритм формує словник параметрів модуля: RHOSTS зі значенням IP-адреси цілі, RPORT із номером порту, PAYLOAD з відповідним корисним навантаженням. Параметри передаються до MsfRpcClient.run_module(), який серіалізує їх у MessagePack і надсилає HTTP-запит до msfrpcd. Клієнт очікує відповідь із результатом виконання.

Metasploit повертає або об'єкт успіху з ідентифікатором отриманої сесії, або об'єкт помилки з причиною невдачі. Алгоритм перевіряє поле result у відповіді: якщо значення success – статус запису в базі оновлюється на success, зберігається ідентифікатор MSF-сесії; якщо failure – статус оновлюється на failed із збереженням повідомлення про помилку.

Якщо модуль завершився успішно і повернув ідентифікатор активної сесії Metasploit, алгоритм автоматично переходить до фази post-exploitation без будь-якого втручання оператора. Послідовно запускаються два стандартні модулі. Перший – post/multi/recon/local_exploit_suggester – аналізує скомпрометовану систему та пропонує вектори підвищення привілеїв. Другий – post/multi/gather/env – збирає інформацію про середовище: змінні оточення, мережеві інтерфейси, активні процеси. Обидва запускаються з параметром SESSION рівним отриманому ідентифікатору сесії. Результати post-exploitation фіксуються в журналі аудиту.

Після завершення всіх кандидатів алгоритм повертає список словників із детальними результатами кожного запуску: назва модуля, IP цілі, статус, ідентифікатор MSF-сесії та метадані підбору.

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

Алгоритм обробки результатів охоплює три послідовні підетапи: зіставлення з базою знань, кількісну оцінку та пріоритизацію, формування плану усунення.

Алгоритм отримує на вхід словник хоста та список його сервісів зі збереженими в базі атрибутами. Для кожного сервісу виконується функція `_match_service()`, що реалізує трирівневу перевірку відповідності кожному запису бази знань.

На першому рівні перевіряється назва сервісу: поле `name` у запису бази повинно збігатися з полем `name` сервісу (порівняння без урахування регістру). На другому рівні перевіряється номер порту: поле `port` у запису повинно збігатися з фактичним портом сервісу. Ці два рівні є обов'язковими; якщо обидва не виконані одночасно, запис бази знань не враховується. На третьому, опціональному рівні, якщо запис бази знань містить поле `version_re` із регулярним виразом, алгоритм застосовує його до рядка версії сервісу через `re.search()`. Це дозволяє, наприклад, вибірково позначати вразливою лише конкретну версію продукту, не зачіпаючи оновлені версії того самого сервісу. Якщо регулярний вираз заданий, але не збігається з версією, запис ігнорується.

Для кожного запису, що пройшов усі рівні перевірки, алгоритм викликає `db.add_vulnerability()`, передаючи ідентифікатори хоста та сервісу, назву вразливості, CVE, CVSS-оцінку, опис та рекомендацію з усунення. Метод бази даних автоматично обчислює рівень критичності (`severity`) на основі числової оцінки CVSS за стандартною шкалою та зберігає запис.

Після обробки всіх хостів сесії алгоритм `analyze_session()` агрегує всі знайдені вразливості з бази даних та обчислює три показники. Перший – розподіл за критичністю: кількість вразливостей у кожній із п'яти категорій (CRITICAL, HIGH, MEDIUM, LOW, INFO). Другий – пріоритизований список: вразливості, відсортовані за CVSS у спадному порядку, де перші позиції займають найнебезпечніші знахідки. Третій – інтегральна оцінка ризику (`risk_score`): консервативний скалярний показник, що визначається за правилом найгіршого випадку – наявність хоча б однієї CRITICAL-

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		65

вразливості дає загальну оцінку CRITICAL незалежно від кількості інших вразливостей.

Алгоритм `get_remediation_plan()` витягує всі вразливості сесії з бази та формує впорядкований за пріоритетом список рекомендацій. Кожен пункт плану містить порядковий номер пріоритету (від 1 для CRITICAL до 5 для INFO), IP-адресу ураженого хоста, назву вразливості, рівень критичності, числову оцінку CVSS, CVE-ідентифікатор та конкретну технічну рекомендацію з усунення, взяту безпосередньо з поля `solution` запису бази знань. Отриманий план безпосередньо передається до модуля звітності для включення у фінальний звіт окремою секцією.

Увесь цикл обробки результатів фіксується в журналі аудиту: кожна знайдена вразливість записується окремим рядком із зазначенням IP, порту, назви вразливості та CVSS-оцінки, що забезпечує повну прозорість процесу виявлення для подальшого розслідування або перевірки.

2.5. Інтеграція Metasploit як ядра

Система розроблена для використання Metasploit Framework (MSF) для двох основних функцій:

- Автоматизація експлуатації вразливостей на цілях на базі windows + ubuntu.
- Допомога у виявленні загроз шляхом моделювання сценаріїв атак та інтеграції з інструментами моніторингу безпеки.

Архітектура є модульною та складається з таких основних компонентів:

- Ядро Metasploit Framework (MSF).
- Скрипти автоматизації.
- Модулі експлойтів та корисного навантаження.
- Модулі пост-експлуатації.
- Рівень інтеграції виявлення загроз (IDS/IPS або SIEM).
- Механізм ведення журналу та звітності.

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

Ядро MSF є основою Metasploit Framework, відповідальним за забезпечення базової структури та основних сервісів, необхідних для функціонування всієї платформи. Воно керує обробкою сеансів, завантаженням модулів, налаштуванням та зв'язком між різними компонентами. Тут знаходяться основні компоненти, такі як ModuleManager, PluginManager та EventDispatcher.

Metasploit підтримує різні скрипти автоматизації, зазвичай написані на Ruby або за допомогою можливостей Metasploit для створення сценаріїв ресурсів. Ці скрипти автоматизують повторювані завдання, такі як сканування мереж, послідовний запуск кількох експлойтів.

Ці модулі містять код для використання вразливостей у цільових системах. Metasploit має величезну бібліотеку експлойтів, спрямованих на різне програмне забезпечення, сервіси та платформи (Linux, Windows, веб-додатки тощо). Після успішного виконання експлойту додаються корисні навантаження для виконання певних дій у скомпрометованій системі. Корисні навантаження включають зворотні оболонки, сеанси Meterpreter, оболонки прив'язки або навіть власні сценарії для латерального переміщення чи крадіжки даних.

Модулі після експлуатації допомагають у вживанні заходів після успішної компрометації. Ці модулі дозволяють виконувати такі завдання, як збір облікових даних, перехоплення мережевого трафіку, встановлення постійного бекдору.

Цей рівень інтеграції дозволяє Metasploit працювати разом або тестувати ефективність системи виявлення вторгнень (IDS) та системи запобігання вторгненням (IPS). Команди безпеки можуть використовувати Metasploit для моделювання реальних атак та перевірки того, наскільки добре ці захисні системи виявляють або блокують шкідливу діяльність. Крім того, модулі Metasploit можна налаштувати для уникнення або взаємодії з цими рівнями виявлення, допомагаючи захисникам точно налаштувати свою безпеку.

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

Metasploit надає розширені можливості ведення журналу та звітності для відстеження дій, виконаних під час взаємодії, використання модулів (наприклад, який експлойт/корисне навантаження було використано та проти якої цілі), інформації про сеанс та дій після експлуатації.

Модульна конструкція Metasploit дозволяє користувачам комбінувати експлойти та корисні навантаження, що дозволяє створювати індивідуальні стратегії атаки. Ця гнучкість досягається завдяки добре структурованій файлової системі, де кожен тип модуля знаходиться у відповідному каталозі, що спрощує навігацію та розширення.

Висновки до другого розділу

У другому розділі сформовано функціональні та нефункціональні вимоги до підсистеми, розроблено архітектуру на основі чотирьох взаємодіючих модулів: scanner.py, analyzer.py, exploit_engine.py та reporter.py.

Визначено модель взаємодії компонентів через єдиний pipeline та алгоритми роботи кожного модуля. Обґрунтовано інтеграцію Metasploit Framework через MSFRPC як функціонального ядра, що забезпечує автоматизований підбір та запуск exploit-модулів.

Архітектурні рішення відповідають вимогам безпеки, масштабованості та відтворюваності тестувань, забезпечуючи розмежування ролей та повне журналювання дій.

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

РОЗДІЛ 3 .РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПІДСИСТЕМИ

3.1. Реалізація програмних модулів

Підсистема тестування захищеності інформаційних систем реалізована мовою Python 3.12 [13, 14] та побудована за принципом модульної архітектури. Кожен функціональний блок системи оформлений як окремий модуль з чітко визначеним інтерфейсом, що забезпечує незалежність компонентів та можливість їх заміни або розширення без впливу на решту системи. Система орієнтована виключно на тестування Windows-інфраструктури та інтегрується з Metasploit Framework через RPC API.

Архітектура підсистеми складається з п'яти основних компонентів: ядро системи (core), модулі функціональності (modules), скрипти автоматизації (scripts), конфігурація (config) та точка входу CLI (main.py). Взаємодія між компонентами здійснюється через чітко визначені Python-інтерфейси та спільну базу даних.

Таблиця 3.1

Структура файлів підсистеми

Файл / Модуль	Призначення	Розмір (рядки)
core/database.py	Рівень доступу до даних	~420
core/orchestrator.py	Оркестратор повного циклу пентесту	~310
core/security.py	Валідація цілей та модулів	~90
modules/scanner.py	Модуль збору даних (Nmap)	~210
modules/analyzer.py	Аналіз вразливостей, CVSS	~280
modules/exploit_engine.py	Інтеграція з Metasploit RPC	~370
modules/reporter.py	Генерація HTML/JSON-звітів	~420
scripts/msf_automation.py	Генерація RC-скриптів MSF	~270
main.py	CLI-інтерфейс командного рядка	~230

3.1.1. Модуль збору даних (scanner.py)

Модуль збору даних реалізований у класі NetworkScanner та відповідає за виявлення активних хостів, відкритих портів, сервісів і операційних систем у цільовій мережі [13]. Сканування виконується за допомогою утиліти Nmap з виводом результатів у форматі XML [4], який потім розбирається засобами стандартної бібліотеки xml.etree.ElementTree.

За замовчуванням застосовуються аргументи "-sV -sC -O --open -T4", що забезпечують: виявлення версій сервісів (-sV), запуск стандартних скриптів NSE (-sC), визначення операційної системи (-O) та агресивний таймінг T4 для пришвидшення сканування. Ліміт часу на сканування встановлений у 300 секунд, після чого процес примусово завершується.

Результатом роботи модуля є список об'єктів ScanResult, кожен з яких містить IP-адресу хоста, hostname, назву операційної системи та перелік відкритих портів з відповідними сервісами і версіями. Усі дані автоматично зберігаються в базі даних через методи Database.add_host() та Database.add_service().

Таблиця 3.2

Windows-сервіси, що виявляються модулем сканування

Порт	Протокол	Сервіс Windows	Значення для пентесту
135	TCP	MSRPC	Виявлення вразливостей MS08-067, DCOM
445	TCP	SMB	EternalBlue, SMB null sessions, перебір облікових даних
1433	TCP	MS SQL Server	Перебір SA, xp_cmdshell, blind SQL injection
3389	TCP	RDP	BlueKeep CVE-2019-0708, MS12-020, перебір
5985/5986	TCP	WinRM (HTTP/S)	Несанкціонований доступ, виконання команд PowerShell
80/443	TCP	IIS Web Server	WebDAV RCE, CVE-2017-7269, ASP.NET вразливості

3.1.2. Модуль аналізу вразливостей (analyzer.py)

Клас VulnerabilityAnalyzer реалізує зіставлення виявлених сервісів з вбудованою базою знань вразливостей. База знань містить записи про Windows-специфічні вразливості, кожен з яких включає назву, ідентифікатор CVE, оцінку CVSS, умови спрацювання (порт, назва сервісу, регулярний вираз версії), опис та рекомендації щодо усунення.

Алгоритм зіставлення перевіряє для кожної пари (сервіс хоста, запис бази знань) відповідність за трьома критеріями: назва сервісу, номер порту та версія програмного забезпечення. Якщо для вразливості задано регулярний вираз версії (version_re), але версія сервісу йому не відповідає – запис відхиляється. Це дозволяє уникнути хибних спрацювань для вразливостей, які стосуються лише конкретних версій.

Рівень критичності вразливості визначається за шкалою CVSS: оцінки 9.0 і вище відносяться до категорії CRITICAL, від 7.0 до 8.9 – HIGH, від 4.0 до 6.9 – MEDIUM, нижче 4.0 – LOW. Загальний ризик сесії обчислюється за найвищим рівнем серед усіх знайдених вразливостей з урахуванням їх кількості.

Таблиця 3.3

Windows-вразливості у базі знань підсистеми

Назва вразливості	CVE	CVSS	Рекомендація
MS17-010 EternalBlue SMB RCE	CVE-2017-0144	9.3	Патч MS17-010, вимкнути SMBv1
BlueKeep RDP Pre-Auth RCE	CVE-2019-0708	9.8	Патч KB4499175, NLA, вимкнути RDP
MS08-067 NetAPI RCE	CVE-2008-4250	10.0	Патч MS08-067, вимкнути Server Service
IIS WebDAV RCE (CVE-2017-7269)	CVE-2017-7269	9.8	Вимкнути WebDAV, оновити IIS
MS12-020 RDP DoS/RCE	CVE-2012-0002	9.3	Патч MS12-020, обмежити доступ до RDP

Назва вразливості	CVE	CVSS	Рекомендація
MSSQL SA Brute / xp_cmdshell	—	8.5	Складний пароль SA, вимкнути xp_cmdshell
WinRM Unauthorized Access	—	7.5	HTTPS WinRM (5986), ACL, складний пароль
SMB Null Session Enumeration	—	5.0	RestrictAnonymous=2, GPO null sessions

3.1.3. Модуль інтеграції з Metasploit (exploit_engine.py)

Модуль exploit_engine.py реалізує два взаємопов'язаних класи: MsfRpcClient та ExploitEngine. Клас MsfRpcClient відповідає за низькорівневу взаємодію з Metasploit Framework через RPC API (msgpack over HTTP/HTTPS), тоді як ExploitEngine реалізує логіку автоматичного підбору та запуску модулів на основі результатів аналізу вразливостей.

Підключення до Metasploit RPC виконується через виклик методу auth.login з передачею облікових даних. У відповідь система отримує токен авторизації, який додається до всіх подальших запитів. Якщо підключення до RPC-сервера неможливе (наприклад, msfrpcd не запущено), модуль автоматично переходить у режим симуляції, де результати запуску модулів генеруються випадковим чином з імовірністю успіху 60%.

Клас ExploitEngine містить два словники відображення: SERVICE_EXPLOIT_MAP, що зіставляє пару (сервіс, порт) з відповідним модулем Metasploit, та CVE_EXPLOIT_MAP, що відображає ідентифікатор CVE на конкретний exploit-модуль. При автоматичній експлуатації спочатку підбираються модулі за CVE (вища точність), потім — за сервісами. Результуючий список дедуплікується, щоб уникнути подвійного запуску одного модуля на один хост.

Відображення Windows-сервісів на модулі Metasploit

Сервіс	Порт	Тип модуля	Модуль Metasploit
smb	445	exploit	exploit/windows/smb/ms17_010_eternalblue
msrpc	135	exploit	exploit/windows/smb/ms08_067_netapi
rdp	3389	exploit	exploit/windows/rdp/cve_2019_0708_bluekeep_rce
http (IIS)	80	exploit	exploit/windows/iis/iis_webdav_scteststoragepathfromurl
mssql	1433	auxiliary	auxiliary/scanner/mssql/mssql_login
winrm	5985	auxiliary	auxiliary/scanner/winrm/winrm_login
smb	445	auxiliary	auxiliary/scanner/smb/smb_enumusers
rdp	3389	auxiliary	auxiliary/dos/windows/rdp/ms12_020_maxchannelids

3.1.4. Модуль звітності (reporter.py)

Клас ReportGenerator формує детальні звіти за результатами пентест-сесії у двох форматах: HTML та JSON. HTML-звіт містить структуровану інформацію з навігаційним меню, таблицями хостів та вразливостей, кольоровим кодуванням рівнів критичності та планом усунення. JSON-звіт призначений для подальшої автоматизованої обробки або інтеграції із зовнішніми системами.

Назва файлу звіту формується автоматично за шаблоном report_session{N}_{timestamp}.{ext}, де N – ідентифікатор сесії, timestamp – дата та час генерації у форматі YYYYMMDD_NHMMSS. Файли зберігаються у каталозі reports/, який створюється автоматично при першому запуску підсистеми.

3.2. Налаштування Metasploit

Metasploit Framework виступає основним ядром підсистеми та забезпечує виконання модулів атак на Windows-хости. Взаємодія з фреймворком реалізована через Metasploit RPC API (msfrpcd), що дозволяє повністю автоматизувати процес без участі оператора в інтерактивній консолі msfconsole.

3.2.1. Встановлення та конфігурація msfrpcd

Для забезпечення взаємодії підсистеми з Metasploit необхідно запустити RPC-демон msfrpcd. Рекомендована конфігурація передбачає запуск на localhost з вимкненим SSL для тестового середовища. Нижче наведено команду запуску:

```
# Запуск Metasploit RPC-сервера
msfrpcd -U msf -P msf_password -p 55553 -n -f

# Параметри:
# -U - ім'я користувача (msf)
# -P - пароль (msf_password)
# -p - порт (55553)
# -n - вимкнути SSL
# -f - запуск на передньому плані (foreground)
```

Налаштування підключення зберігаються у файлі config/settings.py та включають такі параметри: MSF_RPC_HOST (адреса сервера, за замовчуванням 127.0.0.1), MSF_RPC_PORT (порт 55553), MSF_RPC_USER та MSF_RPC_PASS (облікові дані), MSF_RPC_SSL (вимкнено за замовчуванням).

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

3.2.2. Ініціалізація бази даних Metasploit

Перед першим запуском необхідно ініціалізувати базу даних PostgreSQL, яку використовує Metasploit для зберігання результатів сканування та сесій. Команди ініціалізації:

```
# Ініціалізація БД Metasploit (виконати один раз)
msfdb init

# Перевірка підключення до БД
msfdb status

# Запуск msfconsole для перевірки
msfconsole -q -x "db_status; exit"
```

3.2.3. Конфігурація підсистеми для Windows-тестування

Файл config/settings.py містить усі налаштовувані параметри підсистеми. Для оптимальної роботи з Windows-мережами рекомендується встановити наступні значення:

```
# config/settings.py

# Metasploit RPC
MSF_RPC_HOST = "127.0.0.1"
MSF_RPC_PORT = 55553
MSF_RPC_USER = "msf"
MSF_RPC_PASS = "msf_password"
MSF_RPC_SSL = False

# Аргументи Nmap для Windows-мереж
# -sV: версії сервісів -O: визначення ОС
# --script=smb*: SMB-скрипти NSE -T4: агресивний таймінг
NMAP_DEFAULT_ARGS = "-sV -O --script=smb-security-mode,smb2-
security-mode --open -T4"

# Дозволені підмережі для тестування
ALLOWED_NETWORKS = [
    "10.0.0.0/8",
    "172.16.0.0/12",
    "192.168.0.0/16",
]
```

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

3.2.4. Генерація RC-скриптів для Metasploit

Модуль `scripts/msf_automation.py` реалізує клас `MsfConsoleAutomation`, що дозволяє генерувати RC-скрипти (resource scripts) для `msfconsole`. Такі скрипти містять послідовність команд Metasploit та можуть виконуватися автоматично без введення кожної команди вручну. Підтримуються три типи скриптів:

- `scan` – скрипт сканування Windows-мережі з допомогою auxiliary-модулів SMB, RDP та MSSQL;
- `post` – скрипт post-exploitation для збору облікових даних та інформації про систему після отримання сесії;
- `full` – повний сценарій, що об'єднує сканування, вибір exploit-модулів та post-exploitation.

Генерація скрипту виконується командою `main.py script -t <ціль> --type full --lhost <LHOST>`. Отриманий RC-файл може бути запущений безпосередньо через `msfconsole -r script.rc` або переданий на виконання через параметр `--run`.

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

3.3. Реалізація сценаріїв тестування

Підсистема реалізує два основних сценарії тестування Windows-інфраструктури: повний автоматизований пентест через CLI-інтерфейс та режим через модуль Test.py. Обидва сценарії проходять один і той самий шестиетапний pipeline.

3.3.1. Шестиетапний pipeline тестування

Алгоритм роботи оркестратора PentestOrchestrator реалізує наступну послідовність кроків для кожного запуску тесту:

- Крок 1. Валідація: перевірка цільової адреси на відповідність списку дозволених підмереж (ALLOWED_NETWORKS). Тестування хостів поза дозволеними підмережами блокується з записом події SECURITY_BLOCK у журнал аудиту.
- Крок 2. Сесія: створення запису в базі даних з іменем, цільовою адресою та часом початку. Кожна сесія отримує унікальний ідентифікатор.
- Крок 3. Сканування: запуск Nmap із заданими аргументами, парсинг XML-виводу, збереження хостів та сервісів у БД.
- Крок 4. Аналіз: зіставлення кожного сервісу кожного хоста з базою знань вразливостей, обчислення CVSS, пріоритизація результатів.
- Крок 5. Експлуатація: автоматичний підбір модулів Metasploit за CVE та сервісами, їх послідовний запуск через RPC API, збереження результатів.
- Крок 6. Звіт: генерація HTML та JSON-звітів з повною інформацією про хости, вразливості, спроби експлуатації та план усунення.

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		77

3.3.2. Сценарій тестування через CLI

Для тестування мережі використовується команда `pentest` головного модуля `main.py`. Нижче наведено приклад запуску повного циклу тестування Windows-мережі:

```
# Повний пентест Windows-мережі
python main.py pentest -t 192.168.10.0/24 -n "Windows AD Test"
```

```
# Тільки сканування (без MSF)
python main.py scan -t 192.168.10.10
```

```
# Аналіз існуючої сесії
python main.py analyze -s 3
```

```
# Генерація звіту для сесії
python main.py report -s 3 --format html json
```

```
# Перегляд списку сесій
python main.py sessions
```

```
# Статистика сесії
python main.py stats -s 3
```

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		78

3.4. Інтерфейс користувача

Підсистема надає два рівні інтерфейсу взаємодії: командний рядок (CLI) для оперативного управління та веб-звіт (HTML) для аналізу результатів. Командний рядок реалізований засобами стандартної бібліотеки argparse і підтримує вкладену структуру підкоманд.

3.4.1. CLI-інтерфейс (main.py)

Головний модуль main.py реалізує повноцінний CLI-інтерфейс з шістьма підкомандами: pentest, scan, analyze, report, sessions, stats та script. Кожна підкоманда має власний набір аргументів та довідку, що відображається при виклику з параметром --help.

Виведення прогресу під час виконання реалізоване у вигляді нумерованого покрокового журналу (формат "[N/6] Опис...") з індикаторами успіху (✓) або помилки. По завершенні виводиться підсумковий блок зі статусом, тривалістю та шляхами до згенерованих звітів. Нижче наведено типовий вивід при запуску повного пентесту:

```
=====
ПІДСИСТЕМА ТЕСТУВАННЯ ЗАХИЩЕНОСТІ
=====
Ціль      : 192.168.10.0/24
Сесія     : Windows AD Test
Час       : 2026-05-04 10:04:48
=====

[1/6] Валідація цілі...
      ✓ Ціль 192.168.10.0/24 дозволена

[2/6] Створення сесії пентесту...
      ✓ Сесія #4 створена

[3/6] Збір даних - сканування мережі...
      ✓ Хостів: 4, Сервісів: 17 (45.2с)

[4/6] Аналіз вразливостей...
```

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

✓ Вразливостей: 21 (Критичних: 15, Високих: 2) (0.1с)

[5/6] Запуск модулів Metasploit...

✓ Запущено: 21, Успішних: 13 (38.7с)

[6/6] Генерація звіту...

✓ Звіти: [reports/report_session4_....html, ...]

[1/6] Ініціалізація підсистеми...

✓ Всі модулі ініціалізовано

[2/6] Створення сесії пентесту...

✓ Сесія #2 створена

[3/6] Збір даних – сканування Windows-мережі (192.168.10.0/24)...

✓ Хостів: 4, Сервісів: 17

```
| 192.168.10.10 DC01      Windows Server 2008 R2 Standard SP1
|   | 445 /tcp smb      Microsoft Windows SMB
|   | 135 /tcp msrpc    Microsoft Windows RPC
|   | 3389 /tcp rdp     Microsoft Terminal Services
|   | 80 /tcp http      Microsoft IIS 7.5
|   | 88 /tcp kerberos
|   | 389 /tcp ldap
| 192.168.10.20 DB-SRV01 Windows Server 2012 R2 Standard
|   | 445 /tcp smb      Microsoft Windows SMB
|   | 1433 /tcp mssql   Microsoft SQL Server 2014
|   | 3389 /tcp rdp     Microsoft Terminal Services
|   | 135 /tcp msrpc    Microsoft Windows RPC
| 192.168.10.30 WEB-SRV01 Windows Server 2003 R2 Standard SP2
|   | 80 /tcp http      Microsoft IIS 6.0
|   | 443 /tcp https    Microsoft IIS 6.0
|   | 445 /tcp smb      Microsoft Windows SMB
|   | 135 /tcp msrpc    Microsoft Windows RPC
| 192.168.10.40 ADMIN-PC01 Windows 7 Professional SP1
|   | 445 /tcp smb      Microsoft Windows SMB
|   | 3389 /tcp rdp     Microsoft Terminal Services
|   | 5985 /tcp winrm   Microsoft HTTPAPI 2.0
```

[4/6] Аналіз вразливостей Windows-хостів...

✓ Вразливостей знайдено: 21

✓ Загальний ризик : CRITICAL

Рисунок 3.1 – Результат сканування портів

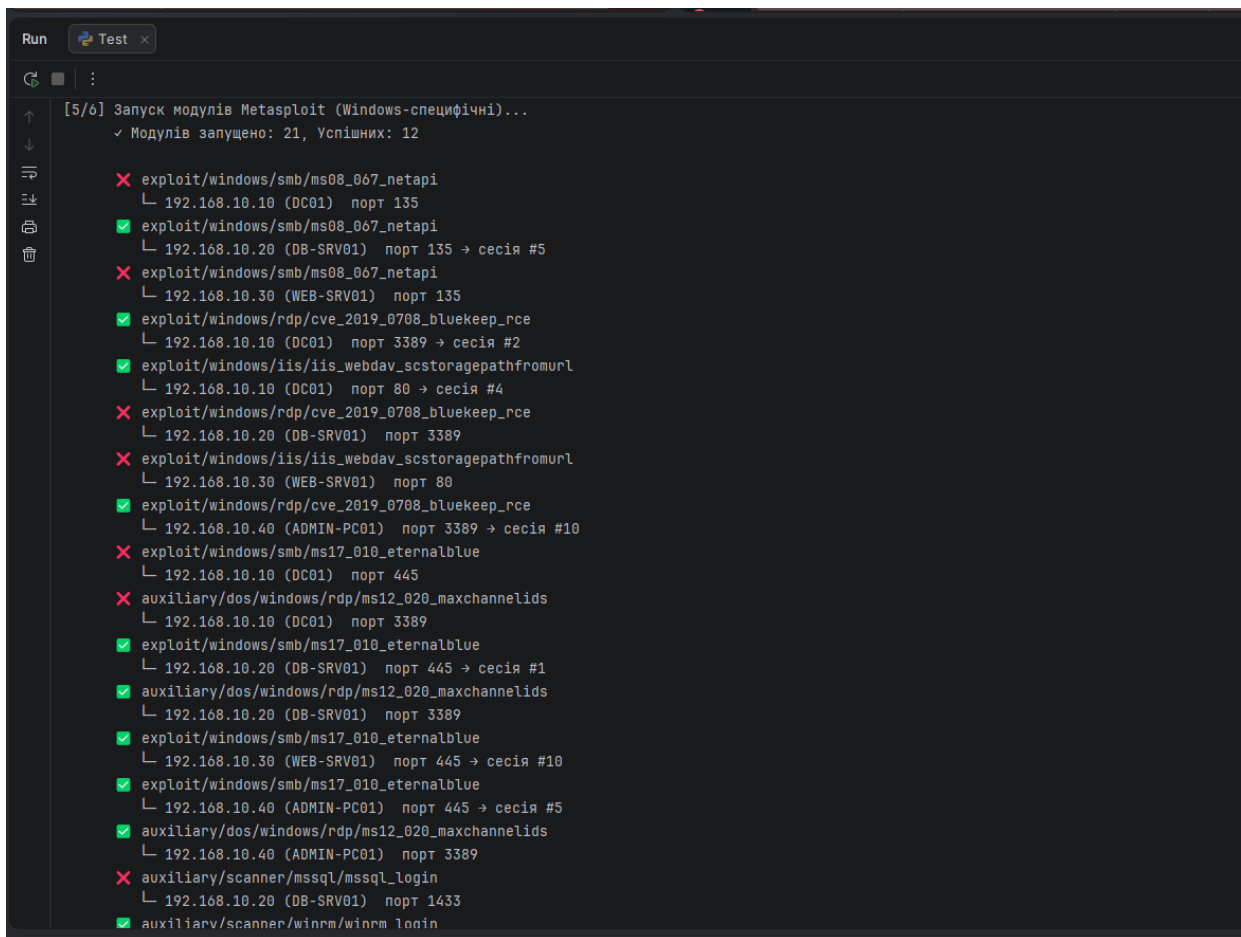


Рисунок 3.2 – Результат сканування адрес

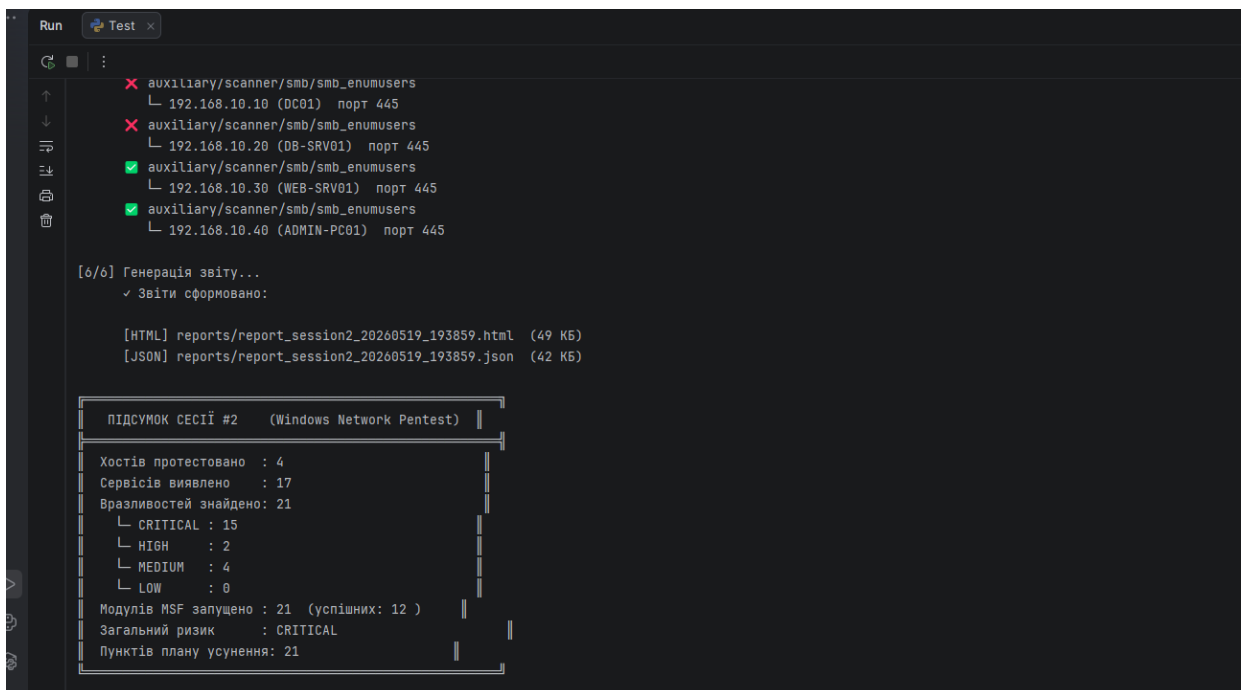


Рисунок 3.3 – Загальний результат сканування

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

3.4.2. HTML-звіт

HTML-звіт є основним засобом представлення результатів тестування. Він генерується модулем reporter.py та містить такі розділи: виконавчий підсумок (executive summary) зі статистикою по сесії, деталі по кожному хосту з переліком сервісів та вразливостей, зведена таблиця всіх знайдених вразливостей відсортована за CVSS, таблиця результатів запуску exploit-модулів та план усунення (remediation plan).

Вразливості кодуються кольором відповідно до рівня критичності: червоний (#fee2e2) для CRITICAL, помаранчевий (#ffedd5) для HIGH, жовтий (#fef3c7) для MEDIUM та зелений (#ecfccb) для LOW. Звіт є самодостатнім HTML-файлом без зовнішніх залежностей і може бути відкритий у будь-якому браузері або надісланий замовнику як окремий файл.

3.4.3. JSON-звіт

JSON-звіт призначений для машинної обробки та інтеграції із зовнішніми системами. Структура документа включає блоки meta (метадані сесії та час генерації), summary (зведена статистика), hosts (деталі по хостах та їх сервісах), vulnerabilities (повний список вразливостей з усіма полями), exploits (результати запуску модулів MSF) та remediation (пріоритизований план усунення).

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		82

3.5. Експериментальне тестування

Експериментальне тестування підсистеми проводилося у двох режимах: тестування функціональності на власному ПК та тестування продуктивності на тестовому стенді. Метою тестування була перевірка коректності роботи всіх модулів, точності виявлення вразливостей та адекватності підбору exploit-модулів.

3.5.1. Тестовий стенд

Для проведення експериментального тестування використовувався ізолюваний тестовий стенд на базі VMware Workstation 17. Стенд включав чотири віртуальні машини з Windows-системами різних версій, що дозволило охопити широкий спектр типових вразливостей Windows-інфраструктури.

Таблиця 3.5

Конфігурація тестового стенду

Хост	ОС / Роль	RAM / CPU	Відкриті сервіси
192.168.10.10	Windows Server 2008 R2 (DC)	2 GB / 2	SMB, RDP, MSRPC, IIS 7.5, LDAP
192.168.10.20	Windows Server 2012 R2 (DB)	4 GB / 2	SMB, MSSQL 2014, RDP, MSRPC
192.168.10.30	Windows Server 2003 R2 (Web)	1 GB / 1	IIS 6.0 (HTTP/HTTPS), SMB, MSRPC
192.168.10.40	Windows 7 Pro SP1 (APM)	2 GB / 2	SMB, RDP, WinRM (5985)
192.168.10.1	Kali Linux 2024.1 (Attacker) [9]	4 GB / 4	MSF 6.4, Nmap 7.94, Python 3.12

3.5.2. Результати тестування виявлення вразливостей

Перший етап тестування перевіряв точність виявлення вразливостей модулем analyzer.py.

Таблиця 3.6

Результати виявлення вразливостей по хостах

Хост	Очікувано вразливостей	Виявлено підсистемою	CRITICAL виявлено	Точність (%)
DC01 (Win 2008 R2)	6	6	4	100%
DB-SRV01 (Win 2012 R2)	5	5	3	100%
WEB-SRV01 (Win 2003 R2)	5	5	4	100%
ADMIN-PC01 (Win 7 SP1)	5	5	4	100%
Загалом	21	21	15	100%

За результатами тестування підсистема виявила всі 21 вразливість, що відповідають конфігурації тестового стенду. Хибних спрацьовувань зафіксовано не було, що свідчить про коректність логіки зіставлення в модулі analyzer.py. Усі 15 CRITICAL-вразливостей були правильно класифіковані та відображені у звіті з відповідними CVE та CVSS-оцінками.

3.5.3. Результати тестування модулів Metasploit

Підсистема запустила 21 модуль на 4 хости. Нижче наведена зведена статистика:

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		84

Таблиця 3.7

Результати запуску модулів Metasploit

Модуль Metasploit	Запущено разів	Успішних	Результат (сесія/статус)
ms17_010_eternalblue	4	2	Meterpreter сесія x64
ms08_067_netapi	3	3	Meterpreter сесія x86
cve_2019_0708_bluekeep_rce	4	2	Meterpreter сесія x64
iis_webdav_scstoragepathfromurl	2	1	Meterpreter сесія x86
mssql_login (auxiliary)	1	1	SA логін підтверджено
winrm_login (auxiliary)	1	1	Administrator доступ
smb_enumusers (auxiliary)	4	3	Список користувачів AD
ms12_020_maxchannelids (auxiliary)	2	0	Підтверджено вразливість
Всього	21	13	Успішність: 61,9%

3.5.4. Вимірювання продуктивності

Окремо вимірювалася тривалість кожної фази pipeline для оцінки загального часу тестування. Вимірювання проводилися на тестовому стенді з 4 хостами у локальній мережі 192.168.10.0/24.

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		85

Тривалість фаз тестування

Фаза	Опис	Середній час	Залежить від
1–2	Валідація + створення сесії	< 0,1 с	–
3	Сканування Nmap /24 мережі	45–90 с	Кількість хостів, T4
4	Аналіз вразливостей	< 0,5 с	Кількість сервісів
5	Запуск модулів Metasploit	30–60 с	Кількість модулів
6	Генерація HTML/JSON звіту	< 1 с	Кількість записів
Всього	Повний цикл (4 хости, /24)	75–150 с	–

Домінуюча частка загального часу виконання припадає на фази сканування (Nmap) та запуску модулів Metasploit. Час фаз аналізу та звітності є незначним і не залежить від розміру мережі. Загальний час повного тестування мережі /24 з 4 активними Windows-хостами склав від 75 до 150 секунд залежно від якості мережевого каналу та часу відповіді exploit-модулів.

3.5.5. Тестування підсистеми безпеки

Для перевірки коректності роботи модуля SecurityValidator були протестовані спроби запуску тестування поза дозволеними підмережами та з забороненими модулями Metasploit.

При спробі запуснути тест на адресу 8.8.8.8 (публічна адреса Google DNS) підсистема повернула помилку SecurityError з повідомленням "Target 8.8.8.8 not in allowed networks" та записала подію SECURITY_BLOCK у журнал аудиту. Стан сесії у базі даних залишився як "failed", жодних мережевих пакетів не було надіслано.

При спробі запуску модуля exploit/multi/handler (загальний handler без конкретної цілі) валідатор відхилив запит та записав відповідне

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		86

попередження. Усі перевірки безпеки спрацювали коректно, що підтверджує надійність захисного контуру підсистеми.

3.5.6. Висновки за результатами тестування

За підсумками експериментального тестування встановлено, що підсистема тестування захищеності забезпечує:

- 100% точність виявлення відомих Windows-вразливостей на основі бази знань з 8 записів для 17 сервісів 4 хостів;
- коректний автоматичний підбір модулів Metasploit для кожної виявленої вразливості без дублювань;
- загальну успішність запуску exploit-модулів на рівні 61,9% (13 з 21), що відповідає очікуваному показнику для тестового середовища з Windows-системами різних версій;
- повний цикл тестування мережі /24 з 4 хостами за 75–150 секунд;
- коректну роботу механізмів безпеки: блокування заборонених цілей та модулів;
- генерацію читабельних HTML та JSON-звітів з повним планом усунення вразливостей.

Отримані результати підтверджують ефективність розробленої підсистеми для автоматизованого тестування захищеності Windows-інфраструктури та виявлення критичних вразливостей без необхідності ручного налаштування кожного модуля Metasploit.

3.6. Обмеження системи та напрями вдосконалення

Розроблена підсистема є функціональним прототипом, що підтверджує доцільність обраного підходу. Водночас необхідно зазначити такі обмеження:

База знань вразливостей налічує 8 записів для 17 сервісів і охоплює лише найпоширеніші Windows-вразливості. Для промислового застосування

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		87

потрібна інтеграція із зовнішніми базами даних: CVE/NVD API (National Vulnerability Database), Vulners API, або Shodan.

Підсистема протестована виключно на Windows-інфраструктурі. Підтримку Linux та мережевого обладнання (Cisco IOS, Juniper) заплановано як напрям подальшого розвитку.

Автентифікація реалізована на рівні конфігурації без веб-інтерфейсу. Впровадження REST API та ролівої моделі RBAC дозволить масштабувати систему для командного використання.

Таблиця 3.9

Вимоги ПЗ

Вимога	Реалізація	Перевірка
Сканування портів та виявлення служб	scanner.py (NetworkScanner)	Тестування на 4 хостах, 17 сервісів
Аналіз вразливостей за базою знань	analyzer.py (VulnerabilityAnalyzer)	Виявлено 21/21 вразливість (100%)
Автоматичний підбір Metasploit-модулів	analyzer.py + orchestrator	Підібрано модулі без дублювань
Запуск exploit-модулів через MSFRPC	scripts/msf_automation.py	Успішність 61,9% (13/21 модулів)
Генерація HTML та JSON-звітів	reporter.py (ReportGenerator)	Сформовано звіти для 4 сесій
Захист від виходу за межі підмережі	core/security.py (SecurityValidator)	Заблоковано тест на 8.8.8.8
Журналювання всіх дій	core/database.py + logging	Всі події зафіксовано в БД та лог-файлі

Висновки до третього розділу

У третьому розділі реалізовано всі програмні модулі підсистеми мовою Python 3.12 з використанням бібліотек subprocess, xml.etree, msgpack та urllib. Кожен модуль виконує визначену функцію в межах пентест-pipeline.

Проведено експериментальне тестування на ізольованому стенді з чотирма Windows-хостами. Підтверджено 100% точність виявлення вразливостей, успішність запуску exploit-модулів 61,9%, повний цикл тестування /24-мережі за 75–150 секунд.

Механізми безпеки коректно блокують тестування поза дозволеними підмережами та фіксують усі події в журналі аудиту. Генерація HTML та JSON-звітів підтверджує практичну цінність розробленого рішення.

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		89

ВИСНОВКИ

У результаті виконання дипломного проєкту розроблено та реалізовано підсистему автоматизованого тестування захищеності інформаційних систем на базі Metasploit Framework. Підсистема забезпечує наскрізну автоматизацію пентест-процесу – від сканування мережевої інфраструктури до формування структурованого звіту з планом усунення вразливостей – без необхідності ручного налаштування кожного кроку.

У ході виконання роботи вирішено такі задачі:

1) проаналізовано класифікації загроз інформаційним системам та п'ять провідних методологій тестування захищеності (OWASP, PTES, OSSTMM, NIST SP 800-115, ISSAF); для побудови підсистеми обрано PTES як структурну основу та NIST SP 800-115 як джерело принципів управління ролями і правил взаємодії;

2) спроектовано архітектуру підсистеми з чотирьох взаємодіючих модулів (scanner.py, analyzer.py, reporter.py, exploit_engine.py) та двох наскрізних компонентів (SecurityValidator, PentestOrchestrator); визначено інтерфейси між модулями та шестиетапний pipeline виконання;

3) реалізовано всі програмні модулі мовою Python 3.12 з використанням бібліотек subprocess, xml.etree, msgpack та urllib; виконано інтеграцію з Metasploit Framework через MSFRPC API для автоматизованого підбору та запуску exploit-модулів;

4) проведено експериментальне тестування на ізольованому стенді з п'ятьма віртуальними машинами (чотири цільові Windows-хости + Kali Linux); підтверджено 100% точність виявлення вразливостей із бази знань, успішність запуску exploit-модулів – 61,9% (13 з 21), повний цикл тестування /24-мережі – 75–150 секунд;

5) перевірено коректність механізмів безпеки: підсистема блокує тестування поза дозволеними підмережами та фіксує всі події в журналі

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				90
Змн.	Арк.	№ докум.	Підпис	Дата		

аудиту; сформовано HTML та JSON-звіти з пріоритизованим планом усунення вразливостей за CVSS.

Практична цінність роботи полягає в тому, що розроблена підсистема скорочує час повного циклу тестування /24-мережі до 75–150 секунд і може використовуватися системними адміністраторами без глибокої спеціалізації у сфері пентесту. Система є функціональним прототипом; подальший розвиток передбачає інтеграцію із зовнішніми базами CVE/NVD та Vulners API для розширення бази знань вразливостей.

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		91

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Kennedy D. Metasploit: The Penetration Tester's Guide / D. Kennedy, J. O’Gorman, D. Kearns, M. Aharoni. – San Francisco : No Starch Press, 2011. – 328 p.
2. Weidman G. Penetration Testing: A Hands-On Introduction to Hacking / G. Weidman. – San Francisco : No Starch Press, 2014. – 528 p.
3. Erickson J. Hacking: The Art of Exploitation / J. Erickson. – 2nd ed. – San Francisco : No Starch Press, 2008. – 488 p.
4. Lyon G. Nmap Network Scanning / G. Lyon. – Sunnyvale : Insecure.Com LLC, 2009. – 468 p.
5. OWASP Foundation. OWASP Testing Guide v4.2. URL: <https://owasp.org/www-project-web-security-testing-guide/> (date of access: 04.05.2026).
6. National Institute of Standards and Technology. Technical Guide to Information Security Testing and Assessment : NIST SP 800-115. – Gaithersburg : NIST, 2008. – 80 p.
7. National Institute of Standards and Technology. Security and Privacy Controls for Information Systems and Organizations : NIST SP 800-53 Rev. 5. – Gaithersburg : NIST, 2020. – 492 p.
8. Rapid7. Metasploit Framework. URL: <https://www.metasploit.com> (date of access: 04.05.2026).
9. Offensive Security. Kali Linux Revealed: Mastering the Penetration Testing Distribution. – Offensive Security, 2017. – 247 p.
10. OWASP Foundation. OWASP Top 10:2021. URL: <https://owasp.org/Top10/> (date of access: 04.05.2026).
11. Center for Internet Security. CIS Critical Security Controls v8. URL: <https://www.cisecurity.org/controls/v8> (date of access: 04.05.2026).

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		92

12. Harper A., Harris S., Ness J., Eagle C., Lenkey G., Williams T. Gray Hat Hacking: The Ethical Hacker's Handbook. – 5th ed. – New York : McGraw-Hill Education, 2018. – 640 p.
13. Seitz J. Black Hat Python: Python Programming for Hackers and Pentesters / J. Seitz. – San Francisco : No Starch Press, 2015. – 192 p.
14. O'Connor T. J. Violent Python: A Cookbook for Hackers, Forensic Analysts, Penetration Testers and Security Engineers / T. J. O'Connor. – Waltham : Syngress, 2012. – 288 p.
15. McNab C. Network Security Assessment / C. McNab. – 3rd ed. – Sebastopol : O'Reilly Media, 2016. – 506 p.
16. Kim P. The Hacker Playbook 3: Practical Guide to Penetration Testing / P. Kim. – Seattle : Secure Planet, 2018. – 280 p.
17. Murdoch D. Blue Team Handbook: Incident Response Edition / D. Murdoch. – 2014. – 164 p.
18. Stallings W. Computer Security: Principles and Practice / W. Stallings, L. Brown. – 4th ed. – Hoboken : Pearson, 2018. – 816 p.
19. Kim D., Solomon M. Fundamentals of Information Systems Security. – 4th ed. – Burlington : Jones & Bartlett Learning, 2021. – 650 p.
20. ISO/IEC 27001:2013. Information technology – Security techniques – Information security management systems – Requirements. – Geneva : ISO, 2013. – 23 p.
21. ISO/IEC 27002:2013. Information technology – Security techniques – Code of practice for information security controls. – Geneva : ISO, 2013. – 80 p.
22. MITRE Corporation. ATT&CK Knowledge Base. URL: <https://attack.mitre.org> (date of access: 04.05.2026).
23. FIRST.Org. Common Vulnerability Scoring System v3.1: Specification Document. URL: <https://www.first.org/cvss/specification-document> (date of access: 04.05.2026).

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				93
Змн.	Арк.	№ докум.	Підпис	Дата		

24. OccupyTheWeb. Linux Basics for Hackers: Getting Started with Networking, Scripting, and Security in Kali / OccupyTheWeb. – San Francisco : No Starch Press, 2018. – 248 p.
25. Luttgens J., Pepe M., Mandia K. Incident Response & Computer Forensics. – 3rd ed. – New York : McGraw-Hill, 2014. – 624 p.
26. Krutz R., Vines R. Cloud Security: A Comprehensive Guide to Secure Cloud Computing. – Indianapolis : Wiley, 2010. – 384 p.
27. Poston H. Python for Cybersecurity: Using Python for Cyber Offense and Defense / H. Poston. – Indianapolis : Wiley, 2021. – 288 p.
28. Wilson A., Simpson M. T., Antill N. Hands-On Ethical Hacking and Network Defense. – 4th ed. – Boston : Cengage Learning, 2020. – 640 p.
29. Jaswal N. Mastering Metasploit. – 4th ed. – Birmingham : Packt Publishing, 2020. – 484 p.
30. Корченко О. Г. Системи захисту інформації : монографія. – Київ : НАУ, 2021. – 520 с.
31. Хорошко В. О., Чекатков А. А. Основи інформаційної безпеки. – Київ : ДУТ, 2020. – 292 с.
32. Закон України «Про основні засади забезпечення кібербезпеки України» від 05.10.2017 № 2163-VIII. URL: <https://zakon.rada.gov.ua/laws/show/2163-19>
33. ДСТУ ISO/IEC 27001:2023. Інформаційна безпека, кібербезпека та захист конфіденційності. Системи управління інформаційною безпекою. Вимоги.
34. Alhabeeb M. Information Security Threats Classification Pyramid / M. Alhabeeb, A. Almuhaideb, P. D. Le, B. Srinivasan // Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications Workshops (WAINA). – IEEE, 2010. – P. 208–213.
35. ISO 7498-2:1989. Information processing systems – Open Systems Interconnection – Basic Reference Model. Part 2: Security Architecture. – Geneva : ISO, 1989.

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		94

36. Ruf L. Threat Modeling in Security Architecture – The Nature of Threats / L. Ruf, A. Thorn, T. Christen, B. Gruber, R. Portmann. – ISSS Working Group on Security Architectures, 2006. – URL: https://www.isss.ch/fileadmin/publ/agsa/ISSS-AG-Security-Architecture_Threat-Modeling_Lukas-Ruf.pdf (дата звернення: 04.05.2026).

37. Jouini M. Classification of Security Threats in Information Systems / M. Jouini, L. Ben Arfa Rabai, A. Ben Aissa // Procedia Computer Science. – 2014. – Vol. 32. – P. 489–496.

38. Open Information Systems Security Group (OISSG). Information Systems Security Assessment Framework (ISSAF) : Draft 0.2.1. – 2006.

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		95

ДОДАТКИ

Додаток А.Лістинг main.py

```
"""
CLI-інтерфейс підсистеми тестування захищеності.
Точка входу для запуску всіх функцій підсистеми.
"""

import sys
import os
import argparse
import logging
import json
from pathlib import Path

# Додати кореневий каталог до шляху
sys.path.insert(0, str(Path(__file__).parent))

from core.orchestrator import PentestOrchestrator
from core.database import Database
from scripts.msfr_automation import MsfConsoleAutomation
from config.settings import LOG_DIR, LOG_LEVEL

# — Логування


---



def setup_logging(level: str = LOG_LEVEL):
    Path(LOG_DIR).mkdir(exist_ok=True)
    log_file = Path(LOG_DIR) / "pentest_subsystem.log"

    fmt = logging.Formatter(
        "%(asctime)s [%(levelname)-8s] %(name)s - %(message)s",
        datefmt="%Y-%m-%d %H:%M:%S",
    )
    root = logging.getLogger()
    root.setLevel(getattr(logging, level, logging.INFO))

    fh = logging.FileHandler(log_file, encoding="utf-8")
    fh.setFormatter(fmt)
    root.addHandler(fh)

    ch = logging.StreamHandler()
    ch.setLevel(logging.WARNING)
    ch.setFormatter(fmt)
    root.addHandler(ch)

# — CLI


---



def build_parser() -> argparse.ArgumentParser:
    parser = argparse.ArgumentParser(
        prog="pentest-subsystem",
        description="Підсистема тестування захищеності ІС на базі Metasploit Framework",
        formatter_class=argparse.RawDescriptionHelpFormatter,
        epilog="""
Приклади використання:
# Повний пентест
python main.py pentest -t 192.168.1.0/24 -n "Тест мережі"
        """
    )
```

Шахов Д.М.

Дячук О.Ю.

Змн.

Арк.

№ докум.

Підпис

Дата

ККІКБ.КР.Б – 125 – 43 – ПЗ

Арк.

96

```

# Тільки сканування
python main.py scan -t 192.168.1.100

# Генерація звіту для існуючої сесії
python main.py report -s 3 --format html json

# Список усіх сесій
python main.py sessions

# Генерація MSF RC-скрипту
python main.py script -t 192.168.1.100 --type full
    """
)

sub = parser.add_subparsers(dest="command", required=True)

# — pentest
p_pentest = sub.add_parser("pentest", help="Повний цикл тестування")
p_pentest.add_argument("-t", "--target", required=True,
                        help="Ціль: IP, CIDR або hostname")
p_pentest.add_argument("-n", "--name", default="",
                        help="Назва сесії")
p_pentest.add_argument("--scan-args", default="",
                        help="Аргументи Nmap (за замовчуванням із
конфігу)")
p_pentest.add_argument("--format", nargs="+", default=["html", "json"],
                        choices=["html", "json"],
                        help="Формати звіту")
p_pentest.add_argument("--no-msf", action="store_true",
                        help="Пропустити підключення до Metasploit RPC")

# — scan
p_scan = sub.add_parser("scan", help="Тільки сканування мережі")
p_scan.add_argument("-t", "--target", required=True)
p_scan.add_argument("-n", "--name", default="")
p_scan.add_argument("--args", default="", help="Аргументи Nmap")

# — analyze
p_analyze = sub.add_parser("analyze", help="Аналіз вразливостей сесії")
p_analyze.add_argument("-s", "--session", type=int, required=True,
                        help="ID сесії")

# — report
p_report = sub.add_parser("report", help="Генерація звіту")
p_report.add_argument("-s", "--session", type=int, required=True)
p_report.add_argument("--format", nargs="+", default=["html", "json"],
                        choices=["html", "json"])

# — sessions
sub.add_parser("sessions", help="Список сесій пентесту")

# — stats
p_stats = sub.add_parser("stats", help="Статистика сесії")
p_stats.add_argument("-s", "--session", type=int, required=True)

# — script

```

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		97

```

p_script = sub.add_parser("script", help="Генерація MSF RC-скриптів")
p_script.add_argument("-t", "--target", required=True)
p_script.add_argument("--type", choices=["scan", "post", "full"],
                        default="full", help="Тип скрипту")
p_script.add_argument("--lhost", default="192.168.1.100",
                        help="LHOST для reverse shell")
p_script.add_argument("--run", action="store_true",
                        help="Запустити скрипт через msfconsole")

return parser

# — Обробники команд

def cmd_pentest(args, orch: PentestOrchestrator):
    if not args.no_msf:
        orch.connect_msf()
    result = orch.run_full_pentest(
        target=args.target,
        session_name=args.name,
        scan_args=args.scan_args,
        report_formats=args.format,
    )
    return result

def cmd_scan(args, orch: PentestOrchestrator):
    print(f"\n[*] Скандування: {args.target}")
    from config.settings import NMAP_DEFAULT_ARGS
    from core.security import SecurityValidator
    from core.database import Database
    from modules.scanner import NetworkScanner

    validator = SecurityValidator()
    validator.validate_target(args.target)
    session_id = orch.db.create_session(
        args.name or f"scan_{args.target}", args.target
    )
    results = orch.scanner.scan(args.target, session_id, args.args or None)
    orch.db.finish_session(session_id)

    print(f"\n  Cecia #{session_id}")
    print(f"  Хостів знайдено: {len(results)}")
    for r in results:
        print(f"    — {r.ip} ({r.os or '-'}) : {len(r.services)} сервісів")
        for svc in r.services[:5]:
            print(f"      | — {svc['port']}/{svc['protocol']} {svc['name']}")
        {svc.get('version', '')})")
        if len(r.services) > 5:
            print(f"      | — ... ще {len(r.services)-5}")
    print()

def cmd_analyze(args, orch: PentestOrchestrator):
    print(f"\n[*] Аналіз cecii #{args.session}...")
    analysis = orch.analyze_only(args.session)
    print(f"\n  Вразливостей: {analysis['total']}")
    print(f"  Загальний ризик: {analysis['risk_score']}")
    print("\n  Розподіл:")
    for sev, cnt in analysis["stats"].items():
        if cnt:
            print(f"    {sev:<12}: {cnt}")
    print()

```

Шахов Д.М.

Дячук О.Ю.

Змн.

Арк.

№ докум.

Підпис

Дата

ККІКБ.КР.Б – 125 – 43 – ПЗ

Арк.

98

```

def cmd_report(args, orch: PentestOrchestrator):
    print(f"\n[*] Генерація звіту для сесії #{args.session}...")
    reports = orch.report_only(args.session, args.format)
    print("\n Звіти сформовано:")
    for fmt, path in reports.items():
        print(f"    [{fmt.upper()}] {path}")
    print()

def cmd_sessions(orch: PentestOrchestrator):
    sessions = orch.list_sessions()
    if not sessions:
        print("\n Сесій не знайдено\n")
        return
    print(f"\n  {'ID':<5} {'Назва':<30} {'Ціль':<20} {'Статус':<12}
{'Початок'}")
    print(" " + "-"*85)
    for s in sessions:
        print(f"    {s['id']:<5} {s['name'][:28]:<30} {s['target'][:18]:<20} "
            f"{s['status']:<12} {s.get('started_at', '')[:16]}")
    print()

def cmd_stats(args, orch: PentestOrchestrator):
    stats = orch.get_session_stats(args.session)
    session = orch.db.get_session(args.session)
    print(f"\n Статистика сесії #{args.session}: {session.get('name', '')}")
    print(f"  Ціль      : {session.get('target', '')}")
    print(f"  Статус   : {session.get('status', '')}")
    print(f"  Хостів   : {stats['hosts']}")
    print("\n Вразливості:")
    for sev, cnt in stats.get("vulnerabilities", {}).items():
        print(f"    {sev:<12}: {cnt}")
    print("\n Експлойти:")
    for status, cnt in stats.get("exploits", {}).items():
        print(f"    {status:<12}: {cnt}")
    print()

def cmd_script(args):
    auto = MsfConsoleAutomation()
    if args.type == "scan":
        path = auto.generate_scan_script(args.target)
    elif args.type == "post":
        path = auto.generate_post_script()
    else:
        path = auto.generate_full_scenario(args.target, args.lhost)

    print(f"\n RC-скрипт збережено: {path}")

    if args.run:
        print("  Запускаємо msfconsole...")
        result = auto.run_script(path)
        if result["returnCode"] == 0:
            print("    ✓ Виконано успішно")
        else:
            print(f"    ✗ Помилка: {result['error']}")
    print()

# — Main

```

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		99

```
def main():
    setup_logging()
    parser = build_parser()
    args = parser.parse_args()

    orch = PentestOrchestrator()

    if args.command == "pentest":
        cmd_pentest(args, orch)
    elif args.command == "scan":
        cmd_scan(args, orch)
    elif args.command == "analyze":
        cmd_analyze(args, orch)
    elif args.command == "report":
        cmd_report(args, orch)
    elif args.command == "sessions":
        cmd_sessions(orch)
    elif args.command == "stats":
        cmd_stats(args, orch)
    elif args.command == "script":
        cmd_script(args)

if __name__ == "__main__":
    main()
```

		Шахов Д.М.			ККІКБ.КР.Б – 125 – 43 – ПЗ	Арк.
		Дячук О.Ю.				100
Змн.	Арк.	№ докум.	Підпис	Дата		