

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 виявлення мережеских вто-
ргнень на базі рішень Snort»

Виконав: студент 4-го курсу групи КБ-21-2
спеціальності 125 «Кібербезпека»

 Євген ГАЙДУЧЕНКО

Керівник: старший викладач кафедри КІ та КБ

 Олександра ПОКОТИЛО

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

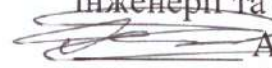
 Олексій ШЕЛУХА

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

3 березня 2025 р.

ЗАВДАННЯ

на випускню кваліфікаційну роботу бакалавра

Студент: Гайдученко Євген Олегович

Тема роботи: «Проект підсистеми виявлення мережових вторгнень на базі рішень Snort»

затверджена Наказом університету від 3 березня 2025 р. №80/с

Термін здачі студентом закінченої роботи 20 червня 2025 р.

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

Об'єкт дослідження: процес забезпечення мережевої безпеки, виявлення мережових вторгнень

Предмет дослідження: методи та засоби виявлення мережових вторгнень, інструменти на базі Snort, а також підходи до їх інтеграції та налаштування в локальній мережі.

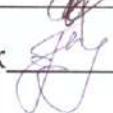
Обладнання та програмне забезпечення:

1. GNS3
2. Oracle VM VirtualBox
3. Kali Linux
4. Ubuntu Linux
5. Snort

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

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Єфіменко А.А	03.03.2025	03.03.2025
2	Єфіменко А.А	14.04.2025	14.04.2025
3	Єфіменко А.А	09.06.2025	09.06.2025

Студент  Євген ГАЙДУЧЕНКО

Керівник  Олександра ПОКОТИЛО

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання	03.03.2025	
2	Аналіз літературних джерел	30.03.2025	
3	Аналіз методів виявлення вторгнень	05.04.2025	
4	Аналіз інструменту Snort	20.04.2025	
5	Проектування мережевої інфраструктури та архітектури системи	25.04.2025	
6	Налаштування мережі та підсистеми виявлення мережевих вторгнень	20.05.2024	
7	Тестування підсистеми виявлення мережевих вторгнень	25.05.2025	
8	Оцінка результатів тестування	30.05.2025	
9	Написання пояснювальної записки	09.06.2025	
10	Отримання відгуку та рецензій		
11	Захист	20.06.2025	

Студент  Євген ГАЙДУЧЕНКО

Керівник  Олександра ПОКОТИЛО

АНОТАЦІЯ

Кваліфікаційна робота на тему «Проект підсистеми виявлення мережових вторгнень на базі рішень Snort» на здобуття освітнього ступеня «Бакалавр» зі спеціальності 125 «Кібербезпека» написана обсягом 51 сторінок і містить 29 ілюстрацій, 6 таблиць та 20 джерел за переліком посилань

Метою кваліфікаційної роботи є комплексне дослідження, реалізація та тестування підсистеми виявлення мережових вторгнень на базі рішень Snort у контексті побудови адаптивного рішення для забезпечення базового рівня кіберзахисту локальної мережі.

Методи дослідження включають аналіз архітектури Snort, тестування з використанням віртуальних машин, застосування сигнатурного та комбінованого аналізу, розробку та узагальнення правил.

Результати дослідження є побудоване тестове середовище в GNS3 з використанням Oracle VM VirtualBox, встановлений та налаштований інструмент Snort, зроблений та налаштований набір правил для виявлення типових атак, а також проведено тестування підсистеми, виявлено особливості пріоритетності спрацювань правил для зниження хибнопозитивних сповіщень.

Протестовано систему виявлення Snort за допомогою віртуальної машини з операційною системою Kali Linux.

Ключові слова: СИСТЕМИ ВИЯВЛЕННЯ ВТОРГНЕНЬ, ВСТАНОВЛЕННЯ ІНСТРУМЕНТУ SNORT, НАЛАШТУВАННЯ ІНСТРУМЕНТУ SNORT, ТЕСТУВАННЯ ІНСТРУМЕНТУ SNORT.

					ККІК.КР.Б – 125 – 04 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Проект підсистеми виявлення мере- жевих вторгнень на базі рішень Snort	Літ.	Арк.	Аркушів
Розроб.		Гайдученко Є.О.		18.06.25				
Керівник		Покотило О.А.		18.06.25			4	50
Реценз.		Шелуха О.О.		18.06.25				
Н. контр.		Покотило О.А.		18.06.25				
Затверд.		Єфіменко А.А.		18.06.25		ФІКТ, Гр. КБ-21-2		

ANNOTATION

The qualification thesis titled "Design of an Intrusion Detection Subsystem Based on Snort Solutions" for the Bachelor's degree in specialty 125 "Cybersecurity" consists of 51 pages and includes 29 illustrations, 6 tables, and 20 references.

The aim of the qualification work is a comprehensive study, implementation, and testing of an intrusion detection subsystem based on Snort solutions, within the context of building an adaptive solution to ensure a basic level of cybersecurity for a local network.

The research methods include analyzing the Snort architecture, testing using virtual machines, applying signature-based and hybrid analysis, and developing and generalizing rules.

The research results include the construction of a test environment in GNS3 using Oracle VM VirtualBox, installation and configuration of Snort, development and configuration of a rule set for detecting typical attacks, as well as subsystem testing. The research also reveals the prioritization features of rule triggers to reduce false-positive alerts.

The Snort detection system was tested using a virtual machine with Kali Linux operating system.

Keywords: INTRUSION DETECTION SYSTEMS, INSTALLATION OF SNORT TOOL, CONFIGURATION OF SNORT TOOL, SNORT TOOL TESTING.

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				5
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ВСТУП.....	8
РОЗДІЛ 1 АНАЛІЗ СИСТЕМ ВИЯВЛЕННЯ МЕРЕЖЕВИХ ВТОРГНЕНЬ	9
1.1 Класифікація системи виявлення вторгнень.....	9
1.2 Типи систем виявлення вторгнень та їх методи	10
1.3 Аналіз системи виявлення вторгнень з відкритим кодом	14
Висновки до розділу	23
РОЗДІЛ 2 СТРУКТУРА СИСТЕМИ ТА ПРАВИЛА ВИЯВЛЕННЯ ВТОРГНЕНЬ НА БАЗІ SNORT.....	24
2.1 Структура системи виявлення вторгнень SNORT	24
2.2. Написання правил Snort	26
2.3 Узагальнення правил за допомогою інверсії	30
2.4 Система ідентифікації та ревізії правил (SID / REV).....	33
Висновки до розділу	36
РОЗДІЛ 3 ДОСЛІДЖЕННЯ ТА РЕАЛІЗАЦІЯ СИСТЕМИ ВИЯВЛЕННЯ	38
ВТОРГНЕНЬ.....	38
3.1 Встановлення програми Snort.....	38
3.2 Налаштування системи Snort.....	40
3.3 Тестування роботи IDS.....	43
Висновки до розділу	47
ВИСНОВКИ.....	48
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	50

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				6
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDS - Intrusion Detection System
NIDS - Network-based Intrusion Detection System
HIDS - Host-based Intrusion Detection System
NNIDS - Node-based Network Intrusion Detection System
AIDS - Anomaly-based Intrusion Detection System
MDS - Misuse Detection System
Snort - Система IDS з відкритим кодом
IP - Internet Protocol
TCP - Transmission Control Protocol
UDP - User Datagram Protocol
ICMP - Internet Control Message Protocol
CLI - Command-Line Interface
GUI - Graphical User Interface
URI - Uniform Resource Identifier
IPsec - Internet Protocol Security
SF - SYN-FIN
DSL - Dynamic Shared Library
JSON - JavaScript Object Notation
SID - Snort ID
ASCII - American Standard Code for Information Interchange
FuzzRule - Компонент генерації узагальнених правил Snort
AlertMerge - Компонент об'єднання журналів у Snort
OS - Operating System
VS Code - Visual Studio Code
Npcap - Бібліотека захоплення мережевого трафіку
WebApp - Web Application

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Системи виявлення вторгнень (IDS) є ключовими елементами у забезпеченні безпеки комп'ютерних мереж та Інтернету, оскільки виконують функції активного захисту. IDS, що працює в режимі реального часу, здатна здійснювати моніторинг мережевого трафіку та активності, оперативно виявляти потенційні загрози, генерувати попередження, зберігати відповідну інформацію в базі даних, а також проводити аналіз інцидентів і створювати журнали вторгнень.

Метою кваліфікаційної роботи є розробка підсистеми виявлення мережевих вторгнень на основі інтеграції та налаштування системи Snort з урахуванням сучасних вимог до мережевої безпеки.

Визначена мета обумовлює наступні завдання:

- дослідити засоби виявлення сучасних мережевих загроз;
- дослідити можливості системи виявлення вторгнень Snort;
- дослідити архітектуру системи виявлення вторгнень із використанням Snort; реалізувати та налаштувати підсистему в тестовому середовищі.

Об'єктом дослідження є процес забезпечення мережевої безпеки, зокрема виявлення та протидія мережевим атакам.

Предметом дослідження є методи та засоби виявлення мережевих вторгнень, зокрема інструменти на базі Snort, а також підходи до їх інтеграції та налаштування в локальній мережі.

У процесі виконання кваліфікаційної роботи було розроблено проект підсистеми виявлення мережевих вторгнень з використанням Snort, яка дозволяє здійснювати моніторинг трафіку, виявляти потенційні загрози та забезпечувати базовий рівень захисту мережі.

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1 АНАЛІЗ СИСТЕМ ВИЯВЛЕННЯ МЕРЕЖЕВИХ ВТОРГНЕНЬ

1.1 Класифікація системи виявлення вторгнень

Система виявлення вторгнень (IDS) являє собою інструмент, призначений для виявлення спроб несанкціонованого доступу до мережі або інформаційних ресурсів шляхом аналізу поведінки користувачів та порівняння її з відомими шаблонами підозрілої активності. Основним завданням IDS є постійний моніторинг мережевого середовища з метою виявлення аномальної, нетипової поведінки, а також потенційних зловживань.

Хоча концепція виявлення вторгнень була відома вже багато років, значна увага до її розвитку й інтеграції в інфраструктуру інформаційної безпеки спостерігається лише останнім часом. Це зумовлено зростанням рівня загроз та потребою у комплексних системах захисту.

IDS аналізує мережевий трафік для виявлення підозрілої активності, і при виявленні потенційної загрози негайно інформує системного або мережевого адміністратора. У деяких випадках система також може автоматично реагувати на інциденти безпеки, наприклад, блокуючи або ізолюючи IP-адресу джерела, що становить загрозу, аби запобігти подальшому доступу до мережі (рис. 1.1).

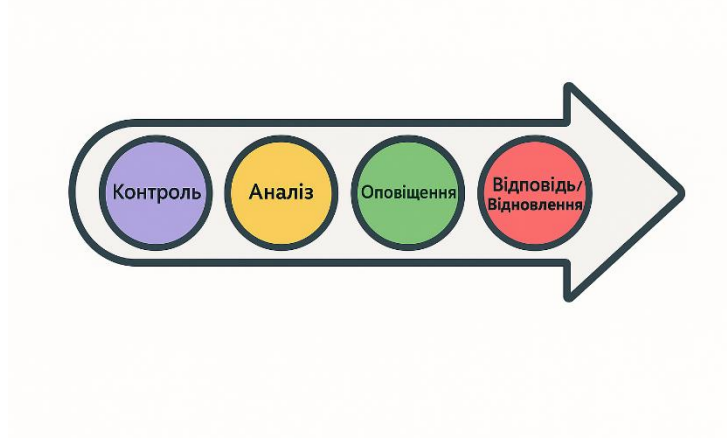


Рисунок 1.1 - Процес IDS

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				9
Змн.	Арк.	№ докум.	Підпис	Дата		

На рисунку 1.1 представлено загальний алгоритм функціонування системи виявлення вторгнень, який охоплює етапи моніторингу, аналізу, оповіщення та реагування на виявлені загрози. Такі системи реалізуються за допомогою спеціального програмного забезпечення, яке здійснює оцінку стану безпеки мережі шляхом безперервного аналізу мережевої активності.

Застосування IDS дозволяє адміністраторам мережі виявляти вразливі місця до того, як ними скористаються потенційні зловмисники. Це забезпечує про активний захист і підвищує загальний рівень інформаційної безпеки.

Усього існує шість основних типів класифікації IDS, які детальніше наведено на рисунку 1.2[1].

HIDS	Host Based (IDS) IDS на основі хоста
NIDS	Network based IDS IDS на основі мережі
HDS	Hybrid IDS Гібридний IDS
NNIDS	Network node intrusion detection Виявлення вторгнень мережевого вузла
AIDS	Anomaly based IDS Ідентифікатори на основі аномалій
MDS	Misuse detection system Система виявлення зловживань

Рисунок 1.2 - Типи IDS

1.2 Типи систем виявлення вторгнень та їх методи

Проте системи виявлення вторгнень розрізняються не лише за архітектурою, а й застосовую різні методи для виявлення шкідливої активності. Розуміння цих методів і їх принципів дозволяє обґрунтовано обрати оптимальну систему під задане середовище чи завдання.

Одним із найпоширеніших методів є сигнатурне виявлення (misuse detection). Цей підхід ґрунтується на порівнянні вхідного трафіку або дій у системі з базою

відомих шаблонів атак (так званих підписів). Якщо поточна активність відповідає відомому візерунку (патерну) загрози, система генерує сповіщення або блокує дію. Сигнатурний метод є надзвичайно ефективним для виявлення вже ідентифікованих атак, таких як: експлойти відомих вразливостей, сканування портів, атаки DoS, SQL-ін'єкції тощо.

Його перевагою є точність і низький рівень хибнопозитивних спрацювань, адже система реагує лише на чітко визначені сигнатури. Водночас основним обмеженням цього методу є його неспроможність виявляти нові або модифіковані атаки, які ще не внесено до бази підписів. Це робить його неефективним у боротьбі з загрозами нульового дня або зі складними цілеспрямованими атаками.

Іншим підходом є аномалійне виявлення, або поведінковий аналіз. У цьому випадку система створює базову модель «нормальної» поведінки користувачів, мережевої активності або системних процесів, і надалі виявляє відхилення від цієї моделі як потенційні загрози. Наприклад, якщо користувач, який зазвичай працює з текстовими документами, раптово запускає інструменти мережевого сканування або здійснює масове копіювання даних, така поведінка може бути класифікована як підозріла. Аномалійні IDS здатні виявляти невідомі загрози, які не мають відповідей у базі сигнатур. Цей підхід реалізується, зокрема, в системах типу AIDS, а також у деяких HIDS-рішеннях.

Основна перевага цього підходу полягає саме в здатності виявляти нові, раніше невідомі атаки. Проте він є більш складним у налаштуванні, оскільки потребує тривалого навчання на реальному трафіку, а також часто супроводжується підвищеним рівнем хибнопозитивних спрацювань, що може перевантажити аналітиків безпеки.

Найбільш сучасним і ефективним є комбінований метод, який інтегрує сигнатурне та аномалійне виявлення. У такій системі підозрілу активність, яка не відповідає жодному з наявних підписів, аналізують з точки зору відхилень від звичної поведінки. Це дозволяє досягти балансу між точністю та гнучкістю, одночасно забезпечуючи ефективне реагування на відомі загрози і виявлення нових.

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				11
Змн.	Арк.	№ докум.	Підпис	Дата		

Реалізація комбінованих систем часто ускладнена через необхідність значних обчислювальних ресурсів і складність координації між модулями, однак їхній рівень захисту значно вищий. Цей метод найбільш часто реалізується у гібридних IDS, де поєднується моніторинг як на рівні хоста, так і мережі. Наприклад, рішення типу Security Onion або сучасні комерційні платформи застосовують саме такий підхід, забезпечуючи всебічний контроль безпеки.

Тепер перейдемо до типів систем.

HIDS (Host-based Intrusion Detection System) призначена для контролю активності на окремому хості, включаючи спроби атаки та несанкціонований доступ до системних ресурсів. Такі системи відслідковують поведінку користувача, доступ до файлів та зміну системних конфігурацій. Деякі сучасні рішення також забезпечують централізоване управління політиками аудиту, аналітику, статистичну обробку подій і можливість контролю доступу. HIDS особливо ефективна для виявлення внутрішніх загроз, оскільки функціонує безпосередньо на рівні операційної системи.

Основним недоліком HIDS є її вразливість у разі компрометації хосту: зловмисник може змінити двійкові файли системи або фальсифікувати журнали подій, що знижує ефективність виявлення[2].

NIDS (Network-based Intrusion Detection System) здійснює аналіз мережевого трафіку, перехоплюючи пакети, що передаються між вузлами мережі. Найчастіше NIDS використовує протокол TCP/IP для зчитування та порівняння пакетів із базою даних відомих загроз (підписів). Це дозволяє виявляти шкідливу активність та атакуючі шаблони на ранніх етапах передачі даних. Також перевіряється і активність пакету, що може свідчити про зловмисну поведінку. Загалом, в порівнянні з іншими типами, система на основі мережі є досить обмеженою.

Традиційні NIDS неефективні в комутованих, зашифрованих або високошвидкісних мережах (понад 100 Мбіт/с). У таких умовах велика частина трафіку може бути прихованою або недоступною для аналізу.

Гібридні IDS поєднують функціональність HIDS і NIDS, що дозволяє централізовано керувати подіями з хостів і мережі одночасно. Такі рішення забезпечують

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				
Змн.	Арк.	№ докум.	Підпис	Дата		12

комплексний захист інформаційної інфраструктури. Наприклад, Cisco реалізувала подібну технологію в комутаторі Catalyst 6000, інтегрувавши механізм виявлення вторгнень у сам пристрій. Компанія ISS, своєю чергою, досягла можливості обробки трафіку на гігабітних швидкостях, що значно розширює сферу застосування гібридних IDS[3].

NNIDS (Node-based Network Intrusion Detection System) призначена для аналізу трафіку вже після його доставки до кінцевого вузла. Це дозволяє виявляти атаки, що проходять повз звичайні NIDS-системи, особливо в зашифрованих або комутованих мережах. NNIDS розміщується на хості і аналізує пакети, які вже досягли системи, подібно до локального сніфера.

Головним обмеженням NNIDS є те, що вона здатна обробляти лише трафік, адресований конкретному вузлу, на якому вона встановлена, на відміну від NIDS, яка може моніторити всю підмережу. Проте в цьому і є головна перевага цієї системи. Вона здатна вирішувати проблеми в складних середовищах де, зазвичай, NIDS не є ефективним.

AIDS (Anomaly-based Intrusion Detection System) здійснює виявлення відхилень від звичайної поведінки користувача або системи. Наприклад, якщо користувач починає використовувати команди або ресурси з частотою, що значно перевищує його звичайний шаблон, система сприймає це як потенційну загрозу.

Ключова перевага AIDS полягає у здатності виявляти нові, невідомі атаки, що не мають підписів у базі. Це досягається завдяки постійному моніторингу поведінки та побудові профілів користувачів[4].

MDS (Misuse Detection System) функціонує на основі порівняння поточної активності з відомими шаблонами атак. Наприклад, багаторазова спроба входу з неправильним паролем може бути класифікована як атака типу «перебір паролів». Система виявлення зловживань є ефективною для ідентифікації вже відомих загроз, однак не здатна виявляти нові або модифіковані атаки, що не відповідають заданим підписам.

Ці системи схожі за принципом дії з антивірусами: вони покладаються на базу візерунків (патернів), яку потрібно постійно оновлювати. Основна проблема

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				
Змн.	Арк.	№ докум.	Підпис	Дата		13

полягає у створенні таких шаблонів, які охоплювали б усі варіанти конкретної атаки, але при цьому не викликали б хибнопозитивних спрацьовувань[5].

1.3 Аналіз системи виявлення вторгнень з відкритим кодом

Snort є програмним забезпеченням з відкритим кодом, створеним Мартіном Решем у 1998 році, яке забезпечує виявлення та запобігання мережевим вторгненням. Це невелика програма, яка функціонує як аналізатор трафіку, реєстратор пакетів або система виявлення та протидії атакам у мережевому середовищі (NIDS). Snort зчитує мережеві пакети та виводить їх на консоль, а за умови активації режиму аналізу -одночасно записує дані на диск у режимі журналювання. Під час спостереження за трафіком, він використовує задані користувачем правила для аналізу та реагування на можливі загрози. (рис.1.3)

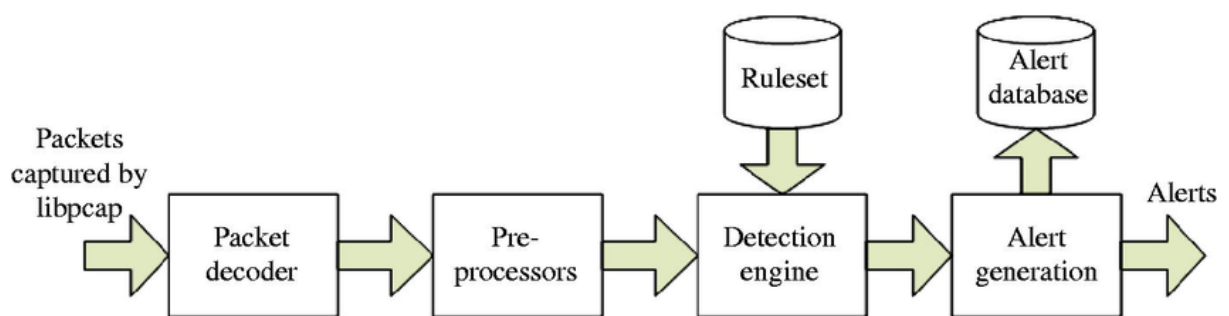


Рисунок 1.3 - Приклад роботи IDS Snort

Snort було створено для використання в операційних системах Linux і Windows, з метою виявлення новітніх атак. Цей інструмент може аналізувати одночасно як мережевий трафік, так і IP-пакети. Крім того, він проводить аналіз протоколів, здійснює пошук вмісту та виконує зіставлення даних. Snort застосовує методи виявлення, засновані на сигнатурах і аномаліях, дозволяючи налаштовувати правила самостійно або використовувати підписи з баз даних на кшталт Emerging Threats[6].

Правила Snort складаються з заголовків і відповідних дій: зокрема, журналювання чи створення сповіщення, вказання мережевого протоколу (наприклад, TCP/UDP/ICMP), IP-адрес джерела та призначення, портів, підписів і класифікаторів (див. таблицю 1.1).

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				
Змн.	Арк.	№ докум.	Підпис	Дата		14

Зразок правил Snort

Елемент	Значення
Дія	alert
Протокол	tcp
Джерело і порт	any any
Призначення і порт	\$HOME_NET any
Повідомлення	"Msg Info: Psybnc Anomaly Access"
Потік	flow:from_server, established
Вміст	content:"Welcome!Psybnc!"
Класифікація	classtype:bad-unknown

У разі, коли виявлено збіг з некоректною сигнатурою, Snort активує правила, визначені користувачем, і виконує відповідну дію. Найбільш поширеними діями є "alert" (створення сповіщення, яке може надсилатися електронною поштою, записуватися в журнал або генерувати спливаюче повідомлення у Windows) та "pass" (ігнорування пакета після збігу з сигнатурою). В таблиці 1.1 показано зразок правила з дією "alert". Це правило спрацьовує, коли вхідний пакет з певним вмістом ("Welcome! Psybnc!") відповідає умовам, і видає попередження адміністратору[7].

Snort вирізняється простотою налаштування і запуску: користувачі можуть самостійно конфігурувати правила. Програма здатна генерувати сповіщення, які обробляються службою Windows Messenger. Хоча інструмент має низку переваг, зокрема зручність управління і гнучкість, серед недоліків варто згадати обмежений графічний інтерфейс, уповільнену роботу при великому трафіку і недостатню підтримку високошвидкісних мереж. Проте, загалом, Snort залишається придатним рішенням для всіх типів застосувань.

Suricata - високопродуктивна система виявлення мережових вторгнень, здатна фіксувати загрози в режимі реального часу. Як і Snort, вона заснована на використанні сигнатур, керованих політиками безпеки та правилами, а також методах виявлення аномалій. Suricata здійснює аналіз мережевого трафіку, застосовуючи гнучку та потужну мову правил, підтримує складні сигнатури й має інтеграцію зі сценаріями, написаними мовою Lua, що дозволяє виявляти комплексні типи атак.

Suricata використовує фільтрацію повідомлень та порогові значення, що охоплюють глобальне та локальне фільтрування, включаючи обмеження швидкості для окремих хостів або підмереж. Побудова правил у Suricata охоплює три основні компоненти: дії, заголовок і параметри правила. Дії можуть бути такими: pass (пропуск), drop (відкидання), reject (відхилення) та alert (сповіщення) [8].

Якщо сигнатура збігається і передбачає дію «pass», Suricata припиняє подальшу перевірку пакета й переходить до аналізу наступного. Дія «drop» зупиняє передачу пакета - така поведінка характерна лише для режиму IPS/inline. У випадку виявлення сигнатури з дією «reject», система не лише блокує пакет, а й надсилає повідомлення про відмову, що, наприклад, у TCP призводить до тайм-аута. Якщо визначено «alert», пакет не блокується, але генерується відповідне повідомлення, яке відображається лише адміністратору.

Ще однією характерною рисою Suricata є параметр напрямку в правилах, який позначається стрілкою вправо (->) і вказує, що збіг буде виявлено лише в пакеті, напрямком якого відповідає зазначеному в сигнатурі.

Таблиця 1.2

Зразок правил Suricata

Правила параметрів	Зразок
Дія	Pass, Drop, alert and Reject
Напрямок	me_PC any \$external_PC any
Параметри правила	Meta-information, headers, payloads and flows

Suricata підтримує ті ж самі набори правил, що й Snort, завдяки чому забезпечується гнучкість у налаштуванні. Водночас ця система має розширені можливості, включаючи багатопоточну обробку та прискорення за допомогою GPU. Однак недоліком є підвищена кількість помилкових спрацювань. Крім того, споживання ресурсів у Suricata є значним, що слід враховувати при виборі системи[9].

Bro IDS - це система виявлення вторгнень, що базується на виявленні аномалій, і зазвичай використовується паралельно зі Snort, оскільки ці два інструменти ефективно доповнюють один одного. В основі Bro лежить доменно-орієнтована мова, створена спеціально для мережевих сценаріїв, на якій реалізовано логіку самої системи. Такий підхід забезпечує гнучке налаштування і робить інструмент

надзвичайно корисним для аналізу трафіку, зокрема в цифровій криміналістиці та подібних галузях.

Архітектура Bro складається з трьох основних компонентів: бібліотеки *libpcap*, що відповідає за захоплення трафіку з мережі; механізму подій (*Event Engine*), який обробляє отримані пакети; та інтерпретатора політик, який виконує скрипти відповідно до встановлених правил. Спочатку *libpcap* приймає весь вхідний трафік з мережевого рівня, відфільтровуючи зайві дані. Далі відфільтровані пакети надсилаються до *Event Engine*, де об'єднуються й аналізуються. Інтерпретатор політик застосовує логіку зі скриптів до кожного пакета, щоб виявити небезпечну або підозрілу активність.

На відміну від *Snort* і *Suricata*, Bro не використовує класичні правила у вигляді фіксованих сигнатур. Натомість він оперує сценаріями, які дають змогу формувати складні поведінкові моделі. Такий підхід забезпечує високу ефективність в аналізі й виявленні загроз у мережах з великою пропускнуою здатністю. Завдяки розподіленій архітектурі Bro здатен рівномірно розподіляти навантаження між кількома серверами, що робить його придатним для масштабованих рішень безпеки.

Bro також здатний виявляти шаблони поведінки, які не виявляються іншими IDS. Проте гнучкість системи має і зворотний бік: налаштування та інтерпретація її роботи вимагають високого рівня технічної підготовки. Для ефективного використання системи необхідні навички програмування, оскільки керування здійснюється через скрипти.

Однією з переваг Bro є можливість адаптації до різних сценаріїв безпеки, оскільки її мова дозволяє реалізовувати як виявлення неправильного використання, так і виявлення аномалій та поведінковий аналіз. Внаслідок цього Bro вирізняється серед IDS високою продуктивністю та гнучкістю, проте не є доцільним в невеликих мережах[10].

OpenWIPS-ng - це система запобігання вторгнень у бездротові мережі з відкритим вихідним кодом, яка побудована за модульною архітектурою. Вона здійснює захоплення бездротового трафіку, виявляє як відкриті, так і приховані мережі, та

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				17
Змн.	Арк.	№ докум.	Підпис	Дата		

ідентифікує можливі спроби вторгнення. Останні дослідження у сфері IDS, зосереджені на налаштуванні таких систем, дозволили адаптувати методи виявлення загроз до певних сегментів мереж, зокрема до специфічних протоколів, які слугують основою для виявлення.

Система складається з трьох основних компонентів:

- Датчик (Sensor) -пристрій, що фіксує бездротовий трафік і передає його на сервер для подальшого аналізу. Також повідомляє про виявлені атаки.
- Сервер (Server) -централізований елемент, який збирає інформацію з усіх підключених датчиків, аналізує її, реагує на загрози, створює відповідні записи у журналах і надсилає сповіщення.
- Графічний інтерфейс користувача (GUI) -призначений для керування сервером, відображення результатів аналізу та інформування про поточні загрози в мережі.

OpenWIPS-ng використовує сигнатурний підхід до виявлення вторгнень і здатен функціонувати на стандартному апаратному забезпеченні. Він виконує основні завдання IDS: сканування мережі, виявлення аномалій та запобігання проникненням.

Система побудована на основі плагінів, завдяки чому до неї можна додавати нові функції за потребою. Для коректної роботи потрібне спеціальне обладнання, проте основною перевагою є здатність блокувати підключення користувачів до небажаних мереж. Система підтримує кілька датчиків одночасно, що суттєво підвищує точність виявлення атак.

Разом з тим, є й недоліки. Система працює лише в межах бездротових середовищ і не підтримує провідні мережі. У випадку помилкового визначення атаки, коли йдеться про легітимного користувача, обидві сторони, користувач і мережа - можуть бути тимчасово роз'єднані. Крім того, трафік між датчиком і сервером не зашифрований, що створює потенційні ризики перехоплення. Хоча система дозволяє адміністратору завантажувати плагіни для розширення функціоналу, вона ціл-

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				18
Змн.	Арк.	№ докум.	Підпис	Дата		

ковито залежить від бездротового підключення. Робота базується на спільних бібліотеках та плагінах, перед запуском яких обов'язково виконуються базові перевірки. Це важливо для оперативного реагування системи при запуску датчиків.

У разі виявлення зловмисника система негайно припиняє процес автентифікації та надсилає відповідне сповіщення адміністратору[11][12].

OSSEC -це система виявлення вторгнень на основі хоста (HIDS), що має відкритий код, високу масштабованість і підтримує кілька платформ. Цей інструмент не зберігає всю інформацію, а лише сповіщення, що знижує навантаження на зберігання. Система має потужний аналітичний механізм з кореляцією подій, вбудованим аналізатором журналів, засобами перевірки цілісності файлів, моніторингом реєстру Windows, централізованим виконанням політик, засобами виявлення rootkit-ів, сповіщенням у реальному часі та активною реакцією на загрози.

Серед можливостей OSSEC - виявлення атак типу «відмова в обслуговуванні» (DoS), простота налаштування, підтримка різних операційних систем і перевірка цілісності системних файлів як у середовищах UNIX, так і у Windows. Важливою перевагою є здатність виконувати контроль цілісності реєстру Windows, що додає гнучкості у середовищах Microsoft.

Однак система має й недоліки. Зокрема, оновлення до нової версії може перезаписати користувацькі правила, замінивши їх стандартними, що потребує обережності під час міграції. Також OSSEC використовує механізм попереднього обміну ключами, який може спричинити труднощі при роботі в режимі сервер-агент у Windows-середовищі.

Система складається з декількох основних компонентів:

- Агент - невелика програма, що встановлюється на захищений хост і передає дані на сервер.
- Сервер - аналізує отримані дані, виконує кореляцію подій і веде журнали безпеки.
- Безагентний режим - дозволяє моніторити файли без встановлення агента, що особливо корисно для брандмауерів, маршрутизаторів та інших Unix-подібних систем.

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				19
Змн.	Арк.	№ докум.	Підпис	Дата		

Система здатна працювати у розподіленому середовищі, обробляючи інформацію з великої кількості джерел, незалежно від того, чи використовується агент чи безагентна модель. Сервер у такій конфігурації відповідає за зберігання логів доступу та аудиту, дозволяючи централізовано керувати безпекою корпоративної мережі[13].

Fragrouter - це утиліта, яка належить до класу NIDS-інструментів і спеціалізується на ухиленні від виявлення в системах виявлення вторгнень. Вона включає набір засобів, що реалізують методи обходу IDS, зокрема інжекцію, ухилення та атаки типу «відмова в обслуговуванні» (DoS), що дозволяє уникати фіксації зловмисної активності.

Інструмент працює на базі протоколів TCP/IP і реалізує механізм односторонньої фрагментації маршрутизованого трафіку. Принцип його дії полягає у тому, що зловмисник надсилає IP-пакети до Fragrouter, який перетворює їх на фрагментований потік даних і пересилає на цільову систему. Завдяки цьому атаки стають менш помітними для стандартних механізмів IDS.

Fragrouter має просту мову правил, яка дозволяє застосовувати до пакетів різні маніпуляції, зокрема затримку, дублювання, скидання, розрив, друк, зміну порядку, сегментацію та зміни вихідного маршруту. Усі ці дії виконуються з пакетами, що надсилаються до хосту-мішені. Приклад синтаксису команди: `fragroute -f dst`.

Головною перевагою Fragroute є відсутність потреби в додаткових бібліотеках, що робить його простим у встановленні та запуску. Він ефективно застосовується для тестування систем IDS, фаєрволів і загалом механізмів захисту TCP/IP-мереж.

Security Onion - це дистрибутив Linux, створений на основі Ubuntu, який об'єднує цілу низку інструментів для виявлення вторгнень та моніторингу мережевої безпеки. Це потужна платформа з відкритим кодом, яка надає інтегроване середовище для IDS, NSM (мережевий моніторинг безпеки) і керування логами.

Security Onion поєднує у собі функціонал таких інструментів, як Snort, Suricata, Bro (тепер Zeek), Sguil, Squert, Snorby, ELSA, Xplico та інші. Це дозволяє

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				20
Змн.	Арк.	№ докум.	Підпис	Дата		

об'єднати кращі характеристики згаданих систем в одному пакеті, створивши універсальне середовище для фахівців з кібербезпеки.

Основні компоненти дистрибутиву:

- Повне захоплення пакетів - здійснюється на всіх інтерфейсах мережі.
- NIDS
- HIDS
- Аналітичні інструменти - забезпечують аналіз трафіку, журналів і подій безпеки.

Security Onion також включає систему сповіщень, дозволяючи адміністраторам миттєво реагувати на потенційні загрози. Це рішення ідеально підходить для побудови інтегрованого середовища моніторингу та контролю безпеки як у малих, так і в великих організаціях[14].

Sguil - це графічна оболонка, що забезпечує зручний інтерфейс для спостереження за подіями в реальному часі, перегляду зафіксованих інцидентів, сесійних даних і збережених мережових пакетів. Вона інтегрується з такими системами, як Snort або Suricata, забезпечуючи ефективний аналіз подій у мережевому середовищі. Завдяки Sguil спеціалісти з безпеки отримують змогу швидко реагувати на загрози, аналізуючи не лише тривоги, а й пов'язані з ними дані трафіку.

Sguil спрощує реалізацію процесу безперервного моніторингу безпеки мережі, дозволяючи операторам ефективно відстежувати, ідентифікувати та досліджувати підозрілі події, що надходять із систем IDS[15].

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				21
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 1.3

Порівняння інструментів виявлення вторгнень

Назва інструменту	Розробник	Тип системи	Опис	Платформи
Snort	Cisco	NIDS, NIPS	Виявляє атаки типу DoS, CGI, сканування портів, SMB та інші. Підтримує одночасний аналіз трафіку й IP-пакетів.	Linux, Windows, кросплатформенна
Suricata	Open Information Security Foundation	NIDS, NIPS	Підтримує автоматичне розпізнавання протоколів, сумісна зі Snort.	Linux, Unix, macOS, Windows
Bro IDS	Vern Paxson	NIDS, AIDS	Працює за сценаріями. Сумісний зі Snort.	Linux, macOS, FreeBSD
OpenWIPS-ng	Aircrack-NG	NIPS	Бездротова модульна система з відкритим кодом.	Linux
Security Onion	-	NIDS	Комплексне середовище, що об'єднує Snort, Suricata, Bro, Sguil, Squert, Xplico та ін.	Linux
OSSEC	Daniel B. Cid	HIDS	Має засоби кореляції, перевірки цілісності, аналізу журналів.	Кросплатформенна
Fragroute	Dug Song	NIDS	Тестує роботу IDS, фаєрволів і TCP/IP-поведінки.	Linux

Таблиця 1.4

Порівняння функціональних характеристик

Параметр	Snort	Suricata	Bro IDS	OpenWIPS-ng	OSSEC	Fragroute
Контекстні сигнатури	—	+	+	—	+	—
Функція IPS	+	+	—	+	—	—
Виявлення DoS	+	+	+	+	+	+

Висновки до розділу

У межах дослідження розглянуто низку відкритих систем IDS, зокрема: Snort, Suricata, Bro (Zeek), OpenWIPS-ng, OSSEC, Fragroute, а також Security Onion, яка об'єднує кілька інструментів.

Усі розглянуті інструменти мають власні особливості й переваги, але з огляду на аналіз функціональних можливостей, методів виявлення загроз, а також ресурсні обмеження і цілі цього дослідження, найбільш доцільним рішенням є Snort.

Ця система забезпечує необхідний рівень захисту на основі сигнатур, є легкою в інсталяції та налаштуванні, а також дозволяє розширення правил і інтеграцію з іншими інструментами.

Саме тому вибір було зроблено на користь Snort, як найоптимальнішого рішення для реалізації системи виявлення вторгнень у даному середовищі.

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				23
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2 СТРУКТУРА СИСТЕМИ ТА ПРАВИЛА ВИЯВЛЕННЯ ВТОРГНЕНЬ НА БАЗІ SNORT

2.1 Структура системи виявлення вторгнень SNORT

У типовій архітектурі Snort можна виділити кілька ключових взаємопов'язаних компонентів, які послідовно обробляють вхідний мережевий трафік.

Перший із них - модуль захоплення трафіку, або сніффер (sniffer), який перехоплює всі вхідні й вихідні пакети в режимі реального часу. Цей модуль може бути розгорнутий у пасивному режимі, коли трафік зчитується з дзеркального порту мережевого пристрою, або у вбудованому (інлайн) режимі, коли система виявлення вторгнень безпосередньо впроваджена у вузол мережі. Захоплений трафік передається далі до модуля розбору пакетів (packet decoder), який відповідає за визначення структури протоколу, вилучення IP-, TCP-, UDP- чи інших заголовків, і передачу даних до наступного етапу.

Після розбору трафік передається препроцесорам (preprocessors), що виконують нормалізацію вхідних даних. Їх основне завдання - уніфікувати формат трафіку, вилучити спроби обходу безпеки шляхом зміни структури пакетів, об'єднати фрагментовані IP-пакети або відновити TCP-сесії. Наприклад, препроцесор Stream5 виконує глибоку перевірку стану TCP-з'єднання, тоді як HTTP Inspect забезпечує розбір HTTP-запитів та усунення маніпуляцій у заголовках, які можуть використовуватись для уникнення детекції.

Центральною частиною архітектури є механізм виявлення (detection engine), який зіставляє нормалізований трафік із набором правил. Правила в Snort мають чітко визначену структуру, що включає тип дії (наприклад, alert, log, drop), протокол (TCP, UDP, ICMP), IP-адресу джерела та призначення, порт, напрямок передачі, а також набір параметрів, які задають умови для спрацювання. Ці параметри можуть включати вміст пакету (content), флаги TCP (flags), напрямок сесії (flow), регулярні вирази (pcre), тип загрози (classtype) та рівень пріоритету (priority). Snort дозволяє створювати складні логічні комбінації з кількох умов, визначаючи контекст, у якому правило повинно активуватись[16].

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				
Змн.	Арк.	№ докум.	Підпис	Дата		24

У випадку, якщо трафік відповідає одному з активних правил, запускається модуль обробки вихідних даних (output plugins), який виконує вказану дію: це може бути створення запису в лог-файл, надсилання повідомлення електронною поштою, виклик зовнішнього скрипта або передача даних у централізовану систему обробки.(рис.2.1)

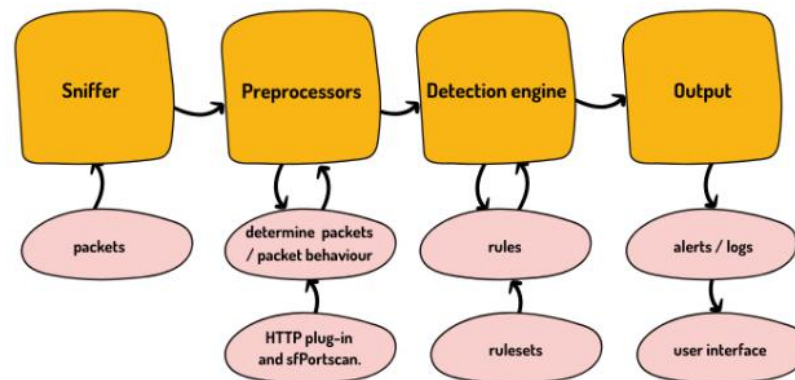


Рисунок 2.1 - Архітектура Snort

Зокрема, Snort підтримує вивід логів у форматі Unified2, JSON або через Syslog, що забезпечує інтеграцію з інструментами аналізу безпеки, такими як SIEM-платформи (Splunk, ELK Stack, Graylog) або проміжні системи, наприклад Barnyard2. Усі повідомлення класифікуються за рівнем пріоритету: від пріоритету 1 (найвища небезпека) до пріоритету 4 (незначна подія).

Крім сигнатурного аналізу, Snort також підтримує поведінкові механізми через складні правила та логіку потоку, що дозволяє фіксувати потенційні аномалії або багатокрокові атаки. Оновлення бази правил можливе вручну або автоматизовано - з використанням сторонніх інструментів, таких як PulledPork, які отримують підписи з відкритих джерел, зокрема Emerging Threats. Таким чином, Snort підтримує актуальність знань про сучасні атаки та здатен оперативно адаптуватись до нових загроз.

Важливо зазначити, що Snort може функціонувати у трьох основних режимах: як звичайний сніффер з виводом пакетів у консоль (Sniffer Mode), як журналювальник трафіку без аналізу (Packet Logger Mode), або як повноцінна система виявлення вторгнень (Network Intrusion Detection Mode), що є основним і найбільш функціональним режимом. У режимі IDS Snort поєднує роботу всіх вищеописаних

компонентів у єдину систему, здатну в режимі реального часу реагувати на шкідливу активність у мережі.

Для підвищення продуктивності Snort дозволяє використовувати апаратні або низькорівневі засоби оптимізації, зокрема бібліотеки DAQ (Data Acquisition Library), технології PF_RING, а також розподіл обробки по ядрах CPU. Особливо актуальним це стає при розгортанні Snort у середовищах з високим трафіком або у гігабітних мережах.

Останні версії Snort (зокрема Snort 3) пропонують оновлену архітектуру з підтримкою багатопоточності, модульної конфігурації через мову Lua, а також YAML-файлів для гнучкого налаштування. Це суттєво розширює можливості системи та підвищує її масштабованість.

Підсумуючи все вищесказане, архітектура Snort включає низку взаємодіючих модулів. У сукупності ці компоненти забезпечують потужну та адаптивну платформу для виявлення широкого спектру мережеских загроз. Така побудова дозволяє ефективно впроваджувати Snort у сучасні інфраструктури безпеки, як у вигляді окремої системи, так і як частину комплексного моніторингового середовища.

2.2. Написання правил Snort

Snort, як і більшість систем мережевого виявлення вторгнень (NIDS), використовує набір сигнатур або підписів для ідентифікації потенційних атак. Ці підписи регулярно оновлюються й публікуються на офіційному сайті проєкту кілька разів на день. Завдяки цьому Snort може вчасно реагувати на нові вразливості та експлойти.

Snort досить гнучкий у способах застосування. Він здатен обробляти як раніше записаний трафік, так і дані з мережі в реальному часі, підтримує широкий вибір форматів виводу, включаючи журнали, бази даних, повідомлення електронною поштою або сторонні системи логування. Завдяки підтримці внутрішніх і сторонніх плагінів Snort здатен працювати у змішаних інфраструктурах із високим навантаженням, при цьому забезпечуючи розширену обробку подій. Snort може працювати з практично будь-якими методами виведення інформації.

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				
Змн.	Арк.	№ докум.	Підпис	Дата		26

Оскільки чітке розуміння правил Snort є важливим для нашого дослідження, наведено детальне пояснення їх структури. Правила Snort відіграють центральну роль у процесі виявлення загроз. Правила - це набір сигнатур, які використовуються для виявлення загроз безпеці. Вони представляють собою набір умов, які зіставляються з мережею у реальному часі або збереженим трафіком. Правила організовані у файли з розширенням .rules. Кожен рядок у такому файлі - окреме правило, яке виконується незалежно. Основний конфігураційний файл snort.conf підключає необхідні файли правил, забезпечуючи їхню активацію для механізму виявлення.

На рисунку 2.2 наведено структуру правила, яка передбачає наявність базової частини та блоку опцій. Базова частина включає дію, протокол, IP-адресу та порт джерела, напрямок трафіку, IP-адресу та порт призначення. Опції уточнюють умови активації правила, вказують на ключові характеристики пакету, визначають повідомлення сповіщення, пріоритет, тип загрози, а також параметри глибини, зсуву, регулярних виразів, сигнатурного вмісту тощо.

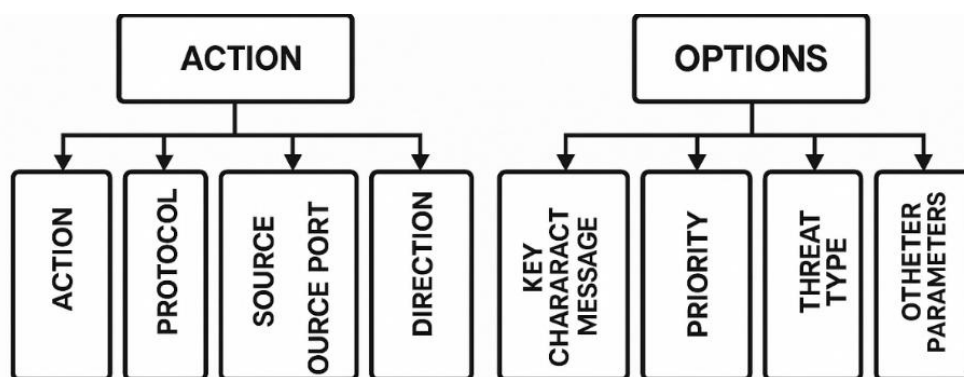


Рисунок 2.2 - Структура правила

Для автоматичного розпізнавання варіантів відомих атак реалізується механізм узагальнення правил. Ідея полягає в тому, що пакет може вважатися потенційно небезпечним, якщо більшість умов правила збігаються, а деякі - частково або з незначними відхиленнями.

Правило Snort має наступний формат:

<дія правила> <протокол> [!] <IP джерела> [!] <порт джерела> <напрямок> <IP призначення> <порт призначення> <опції правила>

- Текст у <кутових дужках> замінюється відповідною обов’язковою змінною (без дужок).

- Текст у [квадратних дужках] є необов’язковим або означає буквальний текст без дужок.

Приклад правила:

alert tcp any any -> any 25

[створити сповіщення для будь-якого вхідного трафіку, що спрямований на порт 25]

Snort повинен згенерувати сповіщення для будь-якого TCP-трафіку, направлено на порт 25, який зазвичай асоціюється з поштовим протоколом SMTP. У даному випадку alert є дією, яку потрібно виконати при збігу (створення сповіщення), tcp - протокол, any any до any 25 - умова, що охоплює будь-яке джерело й IP-призначення, але з фокусом на цільовий порт 25.

Пояснення основних елементів правила:

Дія правила (action): визначає, що робити, якщо умови правила збігаються з пакетом. Зазвичай це створення сповіщення (alert), яке зберігається у файлі або базі даних для подальшого аналізу. Можливі й інші дії, наприклад:

log - запис інформації про пакет без створення сповіщення;

pass - пропуск пакету без аналізу;

activate - активація інших правил або дій.

IP-адреса: може бути задана з використанням формату CIDR (наприклад, 192.168.2.0/12), що позначає діапазон адрес. Для позначення всіх адрес використовується ключове слово any. Перед IP-адресою можна поставити знак оклику !, щоб інвертувати умову (тобто не включати цей діапазон).

Порт: задається як конкретний номер порту, діапазон портів (через двокрапку, наприклад 1:1027), або ключове слово any для всіх портів. Так само можна інвертувати умову знаком !.

Напрямок (direction): визначає напрямок пакета - від джерела до призначення (->) або навпаки (<=).

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				
Змн.	Арк.	№ докум.	Підпис	Дата		28

Опції правила: додаткові параметри, які уточнюють умови спрацювання правила, повідомлення, що буде відображено в сповіщенні, та інші дії. Опції розділяються крапкою з комою, а між ім'ям опції та її значенням стоїть двокрапка.

Застосування узагальнення в цьому випадку полягає у визначенні порогових значень для частоти надходження пакетів, що дозволяє відокремити нормальний ICMP-трафік від аномального. Такий підхід допомагає зменшити кількість помилкових спрацювань та підвищити ефективність виявлення атак типу ICMP spam.

З метою підвищення гнучкості виявлення Snort також підтримує узагальнення правил. Це дозволяє не лише точно відповідати підпису відомої атаки, а й виявляти модифіковані або схожі загрози. В межах експериментального підходу, реалізованого у проєкті FuzzRule, було запропоновано механізм, який автоматично генерує узагальнені варіації існуючих правил. Він базується на двох основних принципах: інверсії параметрів та узагальнення вмісту.

Для кожного оригінального правила створюється резервна копія, після чого генеруються нові варіанти із частковою зміною окремих елементів - наприклад, виключенням певного параметра, зміною напрямку або модифікацією порту. Інверсія, на відміну від видалення, забезпечує логічну коректність співставлення, що критично для збереження достовірності виявлення.

Найбільш ефективними параметрами для узагальнення виявились IP-адреси, порти, напрямки трафіку, протоколи, а також поля вмісту пакету й URI. У разі узагальнення вмісту використовується видалення частини сигнатури з content, або зміщення поля offset та глибини depth, що дозволяє покрити випадки, коли зломисник вносить незначні зміни для уникнення виявлення. Кожному згенерованому правилу може бути присвоєно власне повідомлення із тегами, які вказують на метод модифікації, а також пріоритет, що відображає рівень критичності. Незважаючи на те, що Snort не використовує ці пріоритети напряму, вони полегшують сортування та аналіз подій оператором.

Для інтеграції результатів спрацювань узагальнених правил був реалізований компонент AlertMerge, який поєднує журнали подій, згенеровані як оригінальними, так і новими правилами. Це дозволяє формувати узагальнену аналітичну картину

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				29
Змн.	Арк.	№ докум.	Підпис	Дата		

загроз. Окремий компонент постпроцесора FuzzRule відповідає за статистичний аналіз отриманих сповіщень та виявлення ділянок мережі з підвищеним рівнем хибнопозитивних спрацювань. На основі цих даних адміністратор може коригувати параметри генерації правил, тим самим підвищуючи точність виявлення і знижуючи кількість помилкових сигналів.

Таким чином, механізм створення та узагальнення правил у Snort не лише дозволяє виявляти відомі типи атак, але й адаптувати систему до змін у шаблонах поведінки зловмисників. Відкрита модель правил і гнучкий механізм опцій дозволяють Snort бути високоефективним інструментом мережевого моніторингу, придатним як для великих корпоративних мереж, так і для лабораторних середовищ дослідження безпеки[18][19].

2.3 Узагальнення правил за допомогою інверсії

Під час розробки програмного забезпечення було застосовано метод формування узагальнених правил шляхом видалення одного з параметрів вихідного правила. Наприклад, розглянемо початкове правило:

```
alert udp any any -> any 69 (msg:"TFTP GET Admin.dll"; content: |0001|;
offset:0; depth:2; content:"admin.dll"; offset:2; nocase; classtype:successful-admin;
reference:url,www.cert.org/advisories/CA-2001-26.html; sid:1289; rev:2;)
```

та його узагальнений варіант із вилученим параметром offset:

```
alert udp any any -> any 69 (msg:"TFTP GET Admin.dll"; content: |0001|;
offset:0; content:"admin.dll"; offset:2; nocase; classtype:successful-admin;
reference:url,www.cert.org/advisories/CA-2001-26.html; sid:1289; rev:2;)
```

Видалення параметра offset призводить до того, що більша кількість пакетів буде відповідати узагальненому правилу, порівняно з оригіналом.

Використання даного підходу означає, що будь-який пакет, що задовольняє оригінальне правило, зазвичай також відповідатиме всім його узагальненим варіантам. Враховуючи особливості архітектури SNORT, який створює не більше од-

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				30
Змн.	Арк.	№ докум.	Підпис	Дата		

ного сповіщення на пакет, передбачалося, що система віддаватиме перевагу оригінальному правилу, оскільки воно розташоване перед узагальненими у файлі правил. Проте під час тестування було виявлено, що сповіщення генеруються лише за узагальненими правилами.

Розглянемо приклад правила для виявлення атаки ICMP spam.

ICMP spam - це тип мережевої атаки, який передбачає надсилання великої кількості ICMP-пакетів (наприклад, Echo Request) до цільового вузла з метою перевантаження його мережевого каналу або ресурсів обробки. Така атака є формою відмови в обслуговуванні (DoS), спрямованою на зниження доступності системи або мережі.

Для виявлення подібних атак у системах мережевого моніторингу, таких як SNORT, використовують узагальнені правила, які фіксують повторювані ICMP Echo Request-пакети з різних джерел або з певною частотою, що перевищує нормальний рівень трафіку.

Приклад узагальненого правила SNORT для виявлення ICMP spam може мати вигляд:

```
alert icmp any any -> any any (msg:"Possible ICMP flood detected"; itype:8; threshold:type both, track by_src, count 20, seconds 10; sid:1000001; rev:1;)
```

У цьому правилі:

- itype:8 вказує на ICMP Echo Request - найпоширеніший тип пакету для перевірки доступності (ping).
- Опція threshold задає порогові умови: за останні 10 секунд від одного джерела має надійти щонайменше 20 пакетів.
- msg описує повідомлення, яке з'явиться при спрацюванні правила.
- sid - унікальний ідентифікатор правила.

Детальний аналіз показав, що SNORT організовує правила у вигляді ефектвної структури даних, схожої на бінарне дерево, що оптимізує швидкість перевірки, починаючи з менш складних за умовами правил. Внаслідок цього узагальнені пра-

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				
Змн.	Арк.	№ докум.	Підпис	Дата		31

вила, які містять менше параметрів, мають вищий пріоритет порівняно з оригінальними. Зміна пріоритету або ідентифікатора сповіщень у SNORT не впливає на цю поведінку.

З огляду на це було запропоновано альтернативний підхід - інверсію параметрів правил. Принцип інверсії полягає у визначенні відповідності пакету лише в тих випадках, де схожі умови правила виконуються, але оригінальний параметр не збігається. Відмінність між підходами видалення параметрів і інверсії ілюструється у таблиці 2.1 за допомогою прикладу з чотирма умовами A, B, C і D.

Застосування методу видалення означає, що узагальнене правило може відповідати пакету навіть тоді, коли всі умови оригінального правила збігаються, що призводить до накладення сповіщень. Натомість інверсія гарантує, що при повному збігу оригінального правила жодне узагальнене правило не буде застосоване, що є бажаною поведінкою.

Інверсія параметрів правил ефективно застосовується до більшості типів правил, зокрема інвертування портів, IP-адрес, протоколів, напрямку трафіку та заперечення вмісту. Однак для деяких параметрів узагальнення є складнішим завданням.

Параметри `depth` і `offset`, що визначають область пошуку вмісту в пакетних даних, є прикладом, де узагальнення застосовується шляхом інверсії. Значення `offset` вказує на початкову позицію для пошуку рядка, а `depth` обмежує довжину порівнюваної послідовності.

Варто також відзначити підхід до узагальнення параметра `content`, що часто є ключовим для ідентифікації трафіку. Узагальнення здійснюється шляхом заміни окремих символів рядка вмісту на символ «?», що позначає будь-який символ, а також скороченням рядка для покриття варіацій у початкових або кінцевих символах. Цей метод застосовується до всіх правил із параметрами `content` і `uricontent`.

Компонент `AlertMerge`, схематично представлений на рисунку 2.6, обробляє два файли сповіщень SNORT, згенеровані для одного трафіку: один - з оригінальними, інший - з узагальненими правилами. В результаті створюються три вихідні файли:

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				
Змн.	Арк.	№ докум.	Підпис	Дата		32

- .merged - містить усі сповіщення з оригінального файлу та деякі зі узагальненого;
- .fuzz - містить усі сповіщення з узагальненого файлу, прийняті до .merged;
- .rejected_fuzz - містить узагальнені сповіщення, які не потрапили до .merged.

Цей процес гарантує, що для кожного пакету фіксується лише одне сповіщення, запобігаючи дублюванню через накладання правил.

Постпроцесор FuzzRule підсумовує сповіщення SNORT, незалежно від джерела - оригінального файлу чи файлів AlertMerge. Він формує звіти про загальну кількість кожного типу сповіщень та кількість випадків застосування різних методів узагальнення, що допомагає оцінити ефективність і вплив цих методів на детекцію.

2.4 Система ідентифікації та ревізії правил (SID / REV)

У системах виявлення вторгнень (IDS), таких як Snort, ефективне управління великою кількістю сигнатур є критично важливим для забезпечення якості виявлення та підтримки безпеки. Для цього застосовується система унікальної ідентифікації кожного правила, що складається з двох основних елементів: SID (Signature ID) - ідентифікатора сигнатури, та REV (Revision Number) - номера ревізії правила.

SID (Signature ID) - це унікальний числовий код, призначений для ідентифікації кожного правила у базі Snort. Він грає роль своєрідного "ідентифікаційного номера", що забезпечує унікальність, посилання та інтеграцію.

Унікальність SID полягає в тому, що кожне правило у базі Snort має свій власний, неповторний числовий код. Це дозволяє системі точно ідентифікувати, яке саме правило спрацювало, навіть якщо правила схожі за змістом. Завдяки цій унікальності виключається плутанина, що забезпечує надійність і точність виявлення загроз.

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				
Змн.	Арк.	№ докум.	Підпис	Дата		33

Посилання на SID дає змогу адміністраторам і аналітикам безпеки легко знаходити та обговорювати конкретні правила. Коли в логах або сповіщеннях бачать певний SID, це дозволяє швидко перейти до відповідного правила, що значно спрощує аналіз інцидентів, комунікацію між спеціалістами та налаштування системи. Таким чином, SID стає своєрідним “посиланням” або маркером для точного визначення й обговорення конкретних загроз.

Щодо інтеграції, SID є центральним елементом у взаємодії Snort з іншими системами безпеки - такими як SIEM, системи моніторингу чи автоматичного реагування. Цей унікальний ідентифікатор дозволяє автоматизованим інструментам збирати і аналізувати статистику спрацьовувань правил, відстежувати тренди загроз та організовувати обробку інцидентів. Тобто SID забезпечує зручний і однозначний зв’язок між правилами Snort і зовнішніми системами управління безпекою.

Діапазони SID та їх значення. Для уникнення колізій та забезпечення зручності організації правил, у Snort використовується розмежування SID за діапазонами

Таблиця 2.1

Діапазони SID та їх значення

Діапазон SID	Опис призначення
1 – 999	Внутрішні або тестові правила, зазвичай не використовуються у продуктивних системах
1000 – 99999	Офіційні правила, випущені командою Snort або CERT (Common Event Reporting Team)
100000 – 1,999,999	Користувацькі (custom) правила, створені адміністраторами або сторонніми розробниками
2,000,000 і вище	Резервований діапазон для майбутніх розширень або спеціальних випадків

Ця класифікація дозволяє чітко відокремлювати офіційні правила від користувацьких, що сприяє кращому контролю та обробці оновлень.

Приклад правил:

alert tcp any any -> any 80 (msg:"Example Official Rule"; sid:1001; rev:1;)

alert udp any any -> any 53 (msg:"Custom DNS Query Rule"; sid:150000; rev:1;)

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				
Змн.	Арк.	№ докум.	Підпис	Дата		34

REV (Revision Number) позначає версію конкретного правила. При кожній зміні у сигнатурі (наприклад, корекція умов, додавання або видалення опцій, оптимізація) значення REV збільшується. Це дозволяє відстежувати історію змін правил, легко контролювати оновлення та забезпечувати сумісність, порівнювати різні версії правил, ідентифікувати, яка саме версія правил використовується в конкретній системі.

Якщо потрібно змінити правило, наприклад, виправити помилку або вдосконалити його логіку, то SID при цьому залишається незмінним, а REV збільшується. Це дозволяє чітко відрізнити старі версії правила від нових і уникнути ситуації, коли система вважає різні версії однаковими.

REV також допомагає аналітикам безпеки контролювати, чи використовуються найсвіжіші, перевірені версії правил. Знаючи номер ревізії, можна зрозуміти, наскільки правило актуальне та чи є для нього оновлення. У разі виявлення помилок чи недоліків у правилі, підвищення REV стає сигналом про випуск патчу або оновлення, що полегшує підтримку системи в актуальному і надійному стані.

Приклад оновлення ревізії правил:

```
alert tcp any any -> any 80 (msg:"Example Rule"; content:"malicious";
sid:1000001; rev:1;)
```

```
alert tcp any any -> any 80 (msg:"Example Rule"; content:"malicious"; nocase;
sid:1000001; rev:2;)
```

В наведеному прикладі друга версія правила відрізняється доданою опцією *nocase*, що дозволяє ігнорувати регістр при пошуку рядка "malicious".

У реальному середовищі експлуатації IDS, де адміністратори працюють з великими масивами правил, унікальні SID і контроль REV мають критичне значення. Вони забезпечують точність сповіщень, оскільки виключають дублювання або конфлікти при генерації алертів, що дозволяє системі видавати чіткі і релевантні попередження. Контроль REV допомагає чітко відслідковувати, які правила були оновлені, а які залишилися без змін, що спрощує управління оновленнями і підтримку актуальності системи. (рис.2.3)

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				
Змн.	Арк.	№ докум.	Підпис	Дата		35

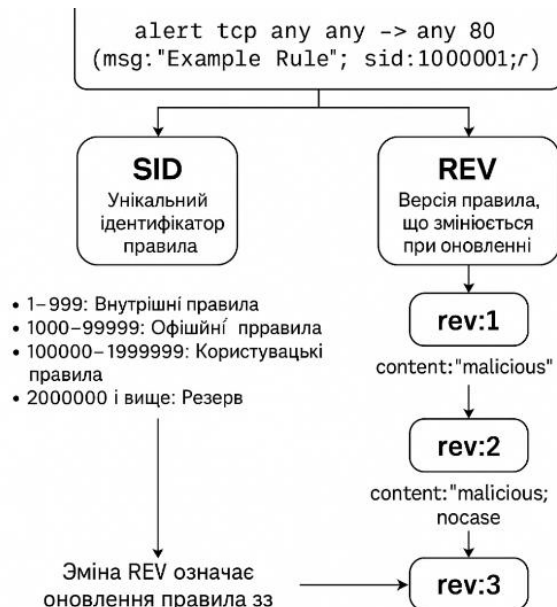


Рисунок 2.3 - Діаграма впливу SID і REV

Для аналізу інцидентів важливо знати не лише яке правило спрацювало, а й його конкретну версію, адже це впливає на розуміння природи загрози та вибір дій для реагування. Нарешті, REV дозволяє автоматизованим системам управління IDS та SIEM ефективно відслідковувати зміни і застосовувати оновлення, що значно покращує якість і швидкість реакції системи безпеки в цілому.

Висновки до розділу

Ретельне налаштування та розробка правил Snort є ключовим фактором ефективності системи виявлення вторгнень. Завдяки сигнатурному підходу, що лежить в основі механізму аналізу, Snort здатен оперативно реагувати на широкий спектр відомих атак. Гнучка структура правил, яка поєднує базові та розширені опції, забезпечує високий ступінь точності детекції.

Серед найбільших переваг Snort - можливість адаптації до змін у шаблонах атак через механізм узагальнення правил. Це дозволяє не лише виявляти атаки з точним збігом підпису, але й модифіковані варіанти, які намагаються обійти класичні засоби виявлення. Зокрема, інверсія параметрів та порогова фільтрація (thresholding) є дієвими підходами для мінімізації хибнопозитивних спрацювань та виявлення аномального трафіку, зокрема ICMP spam або TFTP-атаки.

Втім, структура правил Snort має і низку обмежень. Оскільки основа аналізу - це сигнатурна відповідність, система менш ефективна у виявленні нових, невідомих атак (zero-day) або складних багатокрокових вторгнень без попередньо відомих шаблонів. Крім того, надмірне узагальнення може призвести до зростання кількості хибних спрацьовувань, тоді як занадто вузькі правила - до пропуску потенційних загроз. Тому виникає потреба в балансі між специфічністю та узагальненістю.

Snort також менш ефективний у виявленні атак, що шифруються (наприклад, HTTPS або VPN), оскільки аналіз вмісту стає неможливим без додаткової інтеграції механізмів розшифрування. Складнощі виникають і при виявленні атак на рівні прикладних протоколів, які маскують свій вміст або використовують нестандартні формати запитів.

Загалом, гнучка логіка написання правил Snort, у поєднанні з механізмами узагальнення та інверсії, формує потужну базу для виявлення більшості поширених мережових загроз. Проте ефективне використання системи потребує постійного супроводу, актуалізації правил та аналітичного підходу до їх адаптації під конкретне середовище

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				37
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3 ДОСЛІДЖЕННЯ ТА РЕАЛІЗАЦІЯ СИСТЕМИ ВИЯВЛЕННЯ ВТОРГНЕНЬ

3.1 Встановлення програми Snort

Попередньо, перед встановлення Snort на ОС Ubuntu, потрібно встановити всі необхідні додатки та бібліотеки. Одна з них це “libdaq”. Вона завантажується з [офіційного сайту](#) Snort поряд із необхідною версією.(рис.3.1)

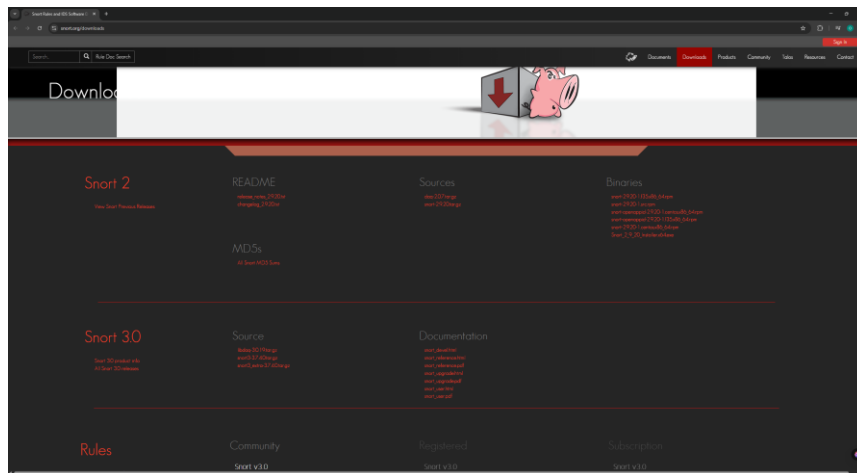


Рисунок 3.1 - Доступні версії Snort

Щоб мати змогу завантажити ширший спектр версій, користувач має зареєструватися на сайті Snort, після чого відкриється персональний кабінет, вигляд якого наведено на рисунку 3.2.

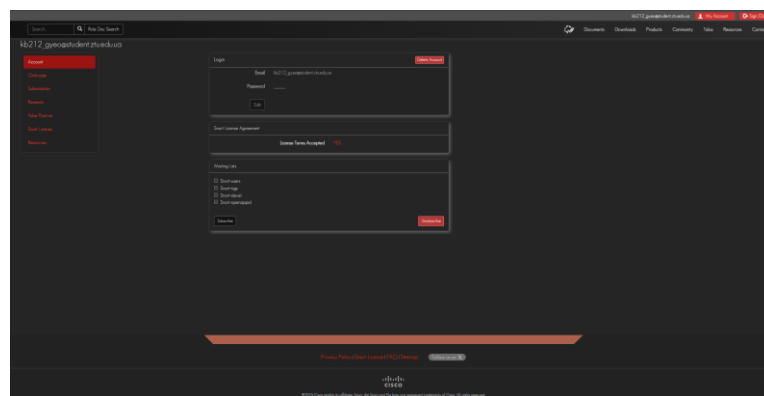


Рисунок 3.2 - Персональний кабінет

Після завантаження додатку, переходимо в Термінал, переходимо в директорию папки Завантажене, бачимо наявність завантаженого архіву та розархівуємо його, як зображено на рисунку 3.3

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				38
Змн.	Арк.	№ докум.	Підпис	Дата		

```
student@student-VirtualBox:~$ cd Завантажене
student@student-VirtualBox:~/Завантажене$ ls
daq-2.0.7.tar.gz
student@student-VirtualBox:~/Завантажене$ tar -zxvf daq-2.0.7.tar.gz
daq-2.0.7/
daq-2.0.7/config.h.in
daq-2.0.7/config.guess
daq-2.0.7/api/
daq-2.0.7/api/daq.h
daq-2.0.7/api/Makefile.am
```

Рисунок 3.3 - Розархівування додатку

Наступним кроком буде конфігурування та встановлення(рис.3.4)

```
student@student-VirtualBox:~/Завантаження$ cd daq-2.0.7/
student@student-VirtualBox:~/Завантаження/daq-2.0.7$ ./configure && make && sudo make instal
l
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
```

Рисунок 3.4 - Конфігурування та встановлення додатку

Лише після встановлення додатку, можна встановити Snort.Адже libdaq необхідний компонент встановлення. Алгоритм аналогічний.(рис.3.5,3.6)

```
student@student-VirtualBox:~/Завантажене$ ls
daq-2.0.7  daq-2.0.7.tar.gz  snort-2.9.16.tar.gz
student@student-VirtualBox:~/Завантажене$ tar xvzf snort-2.9.16.tar.gz
snort-2.9.16/
snort-2.9.16/snort.8
snort-2.9.16/install-sh
snort-2.9.16/snort.pc.in
snort-2.9.16/aclocal.m4
snort-2.9.16/config.guess
snort-2.9.16/compile
snort-2.9.16/config.h.in
snort-2.9.16/missing
```

Рисунок 3.5 - Наявність архіву та його розархівування

		Гайдученко Є.О			<i>ККІКБ.КР.Б – 125 – 04 – ПЗ</i>	Арк.
		Покотило О.А				39
Змн.	Арк.	№ докум.	Підпис	Дата		

```

student@student-VirtualBox:~/Завантажене$ cd snort-2.9.16
student@student-VirtualBox:~/Завантажене/snort-2.9.16$ ./configure --enable-sourcefire && m
ake && sudo make install
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking whether make supports the include directive... yes (GNU style)
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes

```

Рисунок 3.6 - Перехід в каталог з файлами, їх конфігурація та встановлення

Після виконаних нами вище дій, можемо перевірити чи коректно встановився Snort, командою `snort -V`. (рис.3.7) В разі некоректного встановлення, внизу був би відповідний текст.

```

student@student-VirtualBox:~/Завантажене/snort-2.9.20$ cd
student@student-VirtualBox:~$ sudo ldconfig
student@student-VirtualBox:~$ snort -V

_*> Snort! <*-
o"  )~
'   '

Version 2.9.20 GRE (Build 82)
By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
Copyright (C) 2014-2022 Cisco and/or its affiliates. All rights reserved.
Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using libpcap version 1.10.4 (with TPACKET_V3)
Using PCRE version: 8.39 2016-06-14
Using ZLIB version: 1.3

student@student-VirtualBox:~$

```

Рисунок 3.7 - Перевірка встановлення Snort

3.2 Налаштування системи Snort

Після цього потрібно створити необхідні папки командами:

```

sudo mkdir /etc/snort
sudo mkdir /etc/snort/rules
sudo mkdir /etc/snort/rules/iplists
sudo mkdir /etc/snort/preprocrules
sudo mkdir /etc/snort/sorules
sudo mkdir /usr/local/lib/snortdynamicrules
sudo mkdir /var/log/snort
sudo mkdir /var/log/snort/archivedlogs
sudo touch /etc/snort/rules/whitelist.rules
sudo touch /etc/snort/rules/blacklist.rules
sudo touch /etc/snort/rules/local.rules
sudo touch /etc/snort/sid-msg.map

```

Потім, перейти до конфігураційного файлу командою `sudo nano /etc/snort/snort.conf` , та вказати адресу та шляхи до необхідних папок та файлів.(рис.3.8,3.9,3.10)

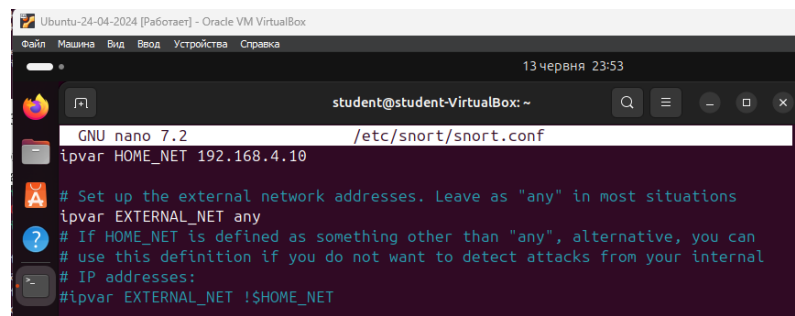
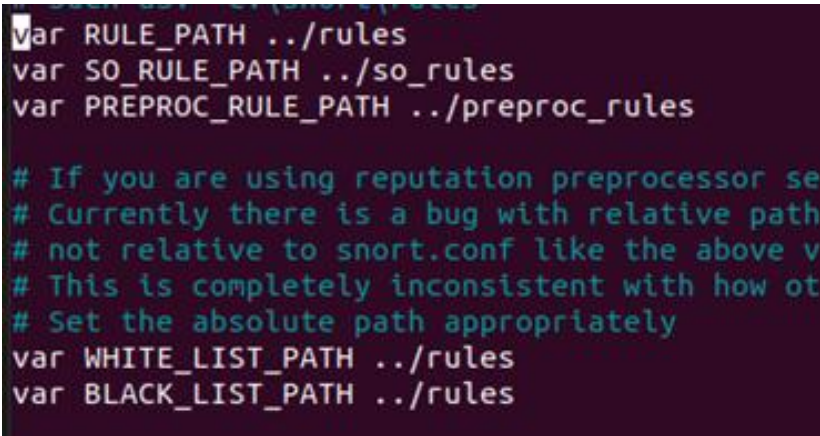


Рисунок 3.8 - Вказана адреса пристрою

За замовчуванням:



Після редагування

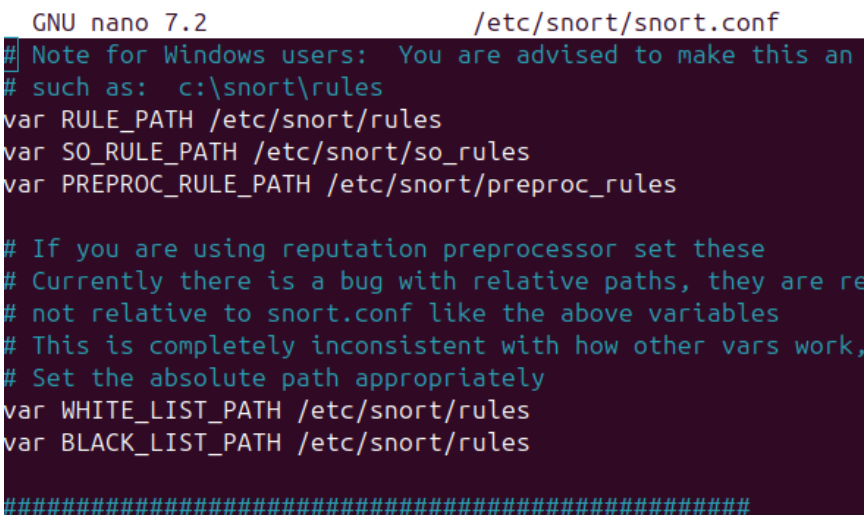


Рисунок 3.9 - Вказані шляхи до файлів

Виконавши це, проведемо тестування конфігурації командою(рис.3.10): `snort -T -i <adpтер> -c /etc/snort/snort.conf`. В нашому випадку, адаптер `enp0s3`

```

student@student-VirtualBox: ~
Using ZLIB version: 1.3

Rules Engine: SF_SNORT_DETECTION_ENGINE Version 3.2 <Build 1>
Preprocessor Object: SF_IMAP Version 1.0 <Build 1>
Preprocessor Object: SF_GTP Version 1.1 <Build 1>
Preprocessor Object: SF_SDF Version 1.1 <Build 1>
Preprocessor Object: SF_DNS Version 1.1 <Build 4>
Preprocessor Object: SF_SSLPP Version 1.1 <Build 4>
Preprocessor Object: SF_SMTP Version 1.1 <Build 9>
Preprocessor Object: SF_S7COMPLUS Version 1.0 <Build 1>
Preprocessor Object: SF_SIP Version 1.1 <Build 1>
Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
Preprocessor Object: SF_POP Version 1.0 <Build 1>
Preprocessor Object: SF_DCERPC2 Version 1.0 <Build 3>
Preprocessor Object: SF_SSH Version 1.1 <Build 3>
Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>
Preprocessor Object: appid Version 1.1 <Build 5>
Preprocessor Object: SF_DNP3 Version 1.1 <Build 1>

Total snort Fixed Memory Cost - MaxRss:104272
Snort successfully validated the configuration!
Snort exiting
student@student-VirtualBox: ~$

```

Рисунок 3.10 - Успішне тестування конфігурації

Для вдалої роботи системи виявлення вторгнень Snort, потрібно написати, або завантажити правила. Далі буде наведено приклад саме написання правил. Для цього відкриваємо файл командою: *sudo nano /etc/snort/rules/local.rules* , і вносимо зміни.(Рис.3.11)

```

student@student-VirtualBox: ~
GNU nano 7.2 /etc/snort/rules/local.rules
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# -----
# LOCAL RULES
# -----
# This file intentionally does not come with signatures. Put your local
# additions here.
alert tcp any any -> $HOME_NET 21 (msg:"FTP connection attempt"; GID:1; sid:1000)
alert icmp any any -> $HOME_NET any (msg:"ICMP test detected"; GID:1; sid:10000)
alert tcp any any -> $HOME_NET 80 (msg:"TELNET connection attempt"; GID:1; sid:10001)

```

Рисунок 3.11 - Написання правил в файл local.rules.

Зберігаємо файл і знову перевіряємо файл конфігурації(рис.3.12)

```

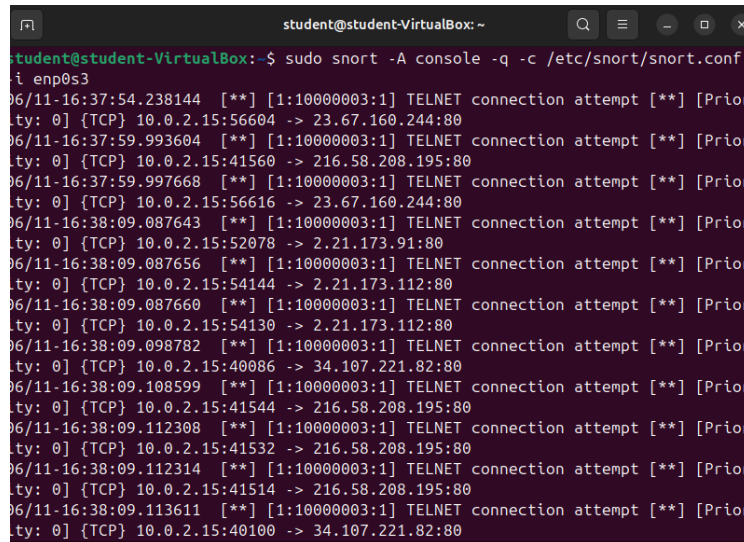
Total snort Fixed Memory Cost - MaxRss:104208
Snort successfully validated the configuration!
Snort exiting
student@student-VirtualBox: ~$

```

Рисунок 3.12 - Перевірка конфігурації після написання правил

3.3 Тестування роботи IDS

Тепер, налаштувавши Snort, можемо запустити систему виявлення вторгнень в режимі мережевих ідентифікаторів командою: `sudo snort -A console -q -u -g -c /etc/snort/snort.conf -i enp0s3` і перевірити його роботу використовуючи браузер.(рис.3.13)



```
student@student-VirtualBox:~$ sudo snort -A console -q -c /etc/snort/snort.conf -i enp0s3
06/11-16:37:54.238144  ** [1:10000003:1] TELNET connection attempt ** [Prior
ty: 0] {TCP} 10.0.2.15:56604 -> 23.67.160.244:80
06/11-16:37:59.993604  ** [1:10000003:1] TELNET connection attempt ** [Prior
ty: 0] {TCP} 10.0.2.15:41560 -> 216.58.208.195:80
06/11-16:37:59.997668  ** [1:10000003:1] TELNET connection attempt ** [Prior
ty: 0] {TCP} 10.0.2.15:56616 -> 23.67.160.244:80
06/11-16:38:09.087643  ** [1:10000003:1] TELNET connection attempt ** [Prior
ty: 0] {TCP} 10.0.2.15:52078 -> 2.21.173.91:80
06/11-16:38:09.087656  ** [1:10000003:1] TELNET connection attempt ** [Prior
ty: 0] {TCP} 10.0.2.15:54144 -> 2.21.173.112:80
06/11-16:38:09.087660  ** [1:10000003:1] TELNET connection attempt ** [Prior
ty: 0] {TCP} 10.0.2.15:54130 -> 2.21.173.112:80
06/11-16:38:09.098782  ** [1:10000003:1] TELNET connection attempt ** [Prior
ty: 0] {TCP} 10.0.2.15:40086 -> 34.107.221.82:80
06/11-16:38:09.108599  ** [1:10000003:1] TELNET connection attempt ** [Prior
ty: 0] {TCP} 10.0.2.15:41544 -> 216.58.208.195:80
06/11-16:38:09.112308  ** [1:10000003:1] TELNET connection attempt ** [Prior
ty: 0] {TCP} 10.0.2.15:41532 -> 216.58.208.195:80
06/11-16:38:09.112314  ** [1:10000003:1] TELNET connection attempt ** [Prior
ty: 0] {TCP} 10.0.2.15:41514 -> 216.58.208.195:80
06/11-16:38:09.113611  ** [1:10000003:1] TELNET connection attempt ** [Prior
ty: 0] {TCP} 10.0.2.15:40100 -> 34.107.221.82:80
```

Рисунок 3.13 - Система виявлення вторгнень в режимі мережевих ідентифікаторів

Можемо спостерігати вивід різних з'єднань що свідчить про правильну роботу системи та правил.

Далі, підключимо до мережі Kali linux та перевіримо роботу правил та системи виявлення вторгнень.(рис.3.14)

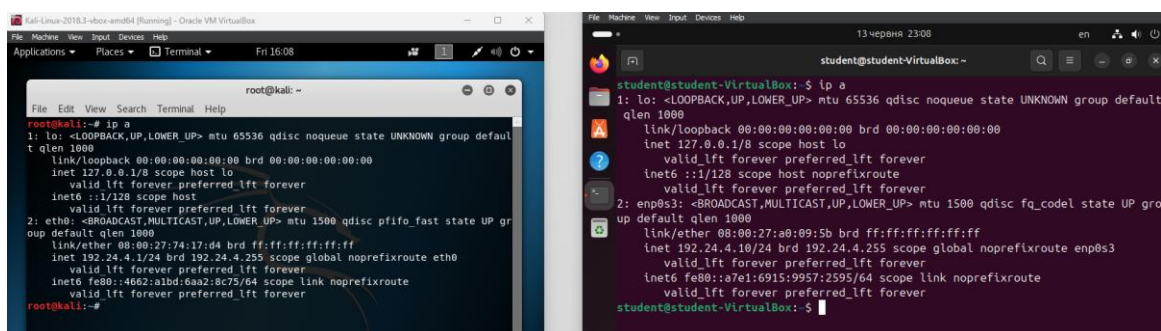


Рисунок 3.14 - Мережеві адреси ОС

Знову запускаємо Snort на Ubuntu, а на Kali Linux виконуємо низку команд для перевірки роботи системи виявлення вторгнень(рис.3.15) ,а саме: `ping 192.168.4.10` (для перевірки реакції на ICMP Test) `nc 192.168.24.10 21` (для перевірки реакції на ftp connection)

nc 192.168.24.10 23 (для перевірки реакції на telnet connection)

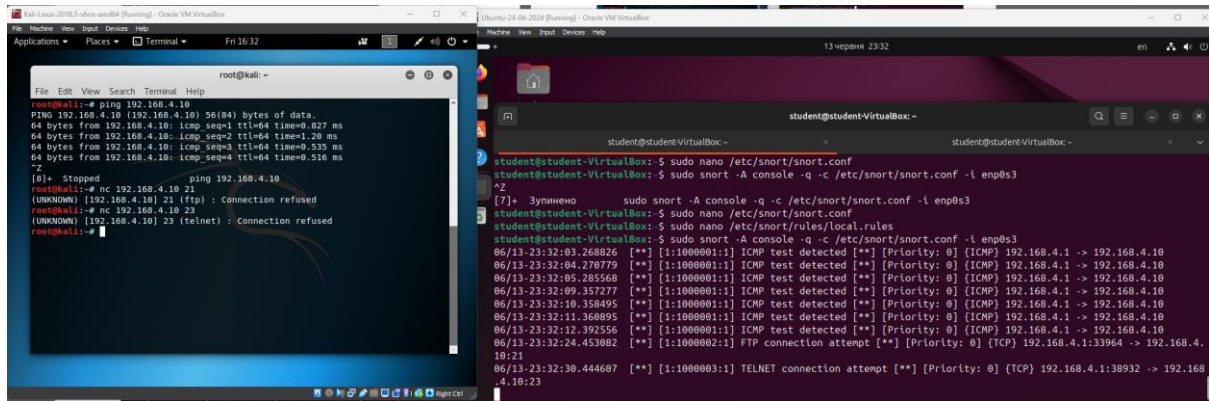


Рисунок 3.15 - Перевірка роботи системи виявлення вторгнень Snort

Загалом, наведено приклад правил які є оптимальним рішенням для більшості мереж, адже цей набір правил задає базовий рівень виявлення активності. Далі будуть наведені приклади можливих розширень та додавання до правил для покращення роботи системи виявлення вторгнень та більш точного виявлення активності.

Перш за все, розглянемо правило для виявлення типового візерунку(патерна) брутфорс-атаки.

alert tcp any any -> any 22 (msg:"Possible SSH Brute-force Attempt"; flags:S; threshold:type threshold, track by_src, count 5, seconds 60; sid:1000004; rev:1;)

Це правило виявляє підозрілу кількість спроб встановлення з'єднання до порту SSH з одного джерела(рис.3.17)

Додаємо правило до файлу local.rules.(рис.3.16)

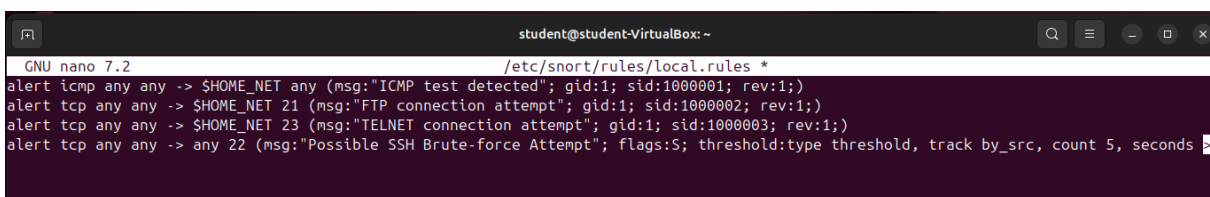


Рисунок 3.16 - Додавання правила

Відкриваємо файл командою *sudo nano /etc/snort/rules/local.rules* , додаємо та зберігаємо. Після чого запускаємо снорт та проводимо атаку з іншої робочої станції.

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				44
Змн.	Арк.	№ докум.	Підпис	Дата		

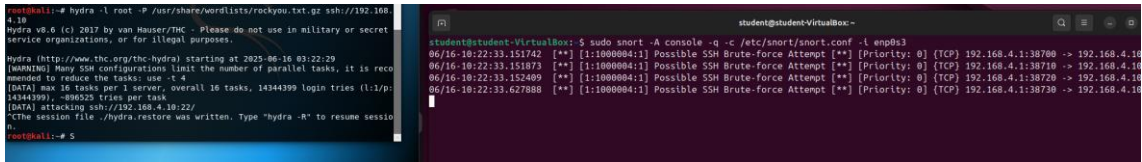


Рисунок 3.17 - Перевірка роботи правила виявлення брутфорс-атаки

Наступним буде правило яке фіксу будь-яке звернення браузера чи до веб-сервера на порту 80.Цей порт є частою ціллю для атак через передачу веб трафіку у відкритому вигляді.(рис.3.20)

alert tcp any any -> any 80 (msg:"HTTP GET Request Detected"; content:"GET"; http_method; sid:1000007; rev:1;)

```
student@student-VirtualBox:~$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.4.1 - - [16/Jun/2025 10:44:11] "GET / HTTP/1.1" 200 -
```

Рисунок 3.18 - Встановлення HTTP-серверу

```
root@kali:~# curl http://192.168.4.10
<!DOCTYPE HTML>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Directory listing for /</title>
</head>
<body>
<h1>Directory listing for /</h1>
<hr>
<ul>
<li><a href=".bash_history">.bash_history</a></li>
<li><a href=".bash_logout">.bash_logout</a></li>
<li><a href=".bashrc">.bashrc</a></li>
```

Рисунок 3.19 - Проведення атаки

```
student@student-VirtualBox:~$ sudo snort -A console -q -c /etc/snort/snort.conf -i enp0s3
[sudo] пароль до student:
06/16-10:44:11.549605  [**] [1:1000007:1] HTTP GET Request Detected [**] [Priority: 0] {TCP
} 192.168.4.1:50880 -> 192.168.4.10:80
```

Рисунок 3.20 - Спрацювання правила

Останнім в списку, але не по важливості, буде правило виявлення підозрілих з'єднань до нестандартного порту. Це часто використовується для "бекдору". Порт оберемо 4444,правила має наступний вигляд:

alert tcp any any -> any 4444 (msg:"Possible Netcat Backdoor Connection"; sid:1000008; rev:1;)

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				45
Змн.	Арк.	№ докум.	Підпис	Дата		

Для перевірки роботи правила нам потрібно відкрити цей порт за допомогою команди `sudo nc -lvp 4444`. (рис.3.21)

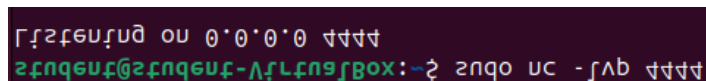


Рисунок 3.21 - Відкриття порту 4444

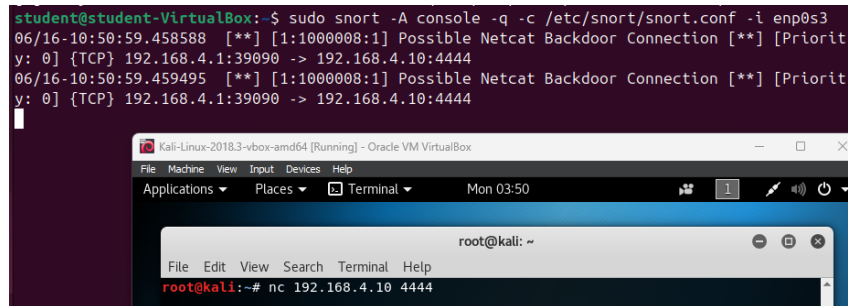


Рисунок 3.22 - Підключення до нестандартного порту та спрацювання правила.

Наведений набір правил Snort охоплює базові сценарії виявлення мережових атак і загроз. Загалом, даний набір правил є ефективним для виявлення типових проявів атак на ранніх стадіях, а також для побудови фундаменту системи виявлення загроз. Проте, правила не здатні розрізнити легітимні та зловмисні HTTP-запити, не перевіряють вміст запитів на ознаки ін'єкцій, і не аналізують відповідь сервера, що залишає простір для вдосконалення.

Удосконалення даного набору правил доцільно проводити в кількох напрямках. По-перше, слід впровадити змісто-орієнтовані правила, які ідентифікують ознаки поширених атак, таких як SQL injection, XSS чи directory traversal. Наприклад, додавання перевірки на характерні візерунки(патерни) в HTTP URI або параметрах, значно підвищить точність виявлення.

По-друге, важливо інтегрувати динамічні механізми фільтрації та кореляції подій, які дозволяють враховувати поведінковий контекст, зокрема часові та просторові залежності між подіями. Це може включати в себе виявлення послідовності атак або накопичення аномалій від одного джерела.

Нарешті, перспективним напрямом є інтеграція Snort з інструментами аналізу логів, SIEM-системами або технологіями машинного навчання, які дозволяють будувати профілі типових сценаріїв мережевої активності та виявляти відхилення в реальному часі.

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				
Змн.	Арк.	№ докум.	Підпис	Дата		46

Висновки до розділу

У даному розділі було здійснено поетапну реалізацію системи виявлення вторгнень (IDS) на основі Snort в операційній системі Ubuntu. Проведено інсталяцію всіх необхідних компонентів, зокрема бібліотеки libdaq, без якої запуск Snort є неможливим. Особливу увагу було приділено правильній послідовності дій при встановленні та конфігурації системи, що забезпечило її коректну роботу.

Налаштування конфігураційних файлів та створення відповідної структури директорій дозволили адаптувати Snort до вимог середовища, у якому він функціонує. Особливо важливим кроком було створення та редагування правил виявлення, оскільки саме вони визначають ефективність IDS у реальних умовах.

Проведене тестування підтвердило працездатність системи та її здатність виявляти типові атаки. Це свідчить про правильне налаштування та інтеграцію всіх складових елементів Snort у загальну систему безпеки. Проаналізовано можливості удосконалення системи та покращення її роботи.

Отже, реалізована система виявлення вторгнень є готовою до базового використання та подальшого розширення її функціоналу. У ході роботи було отримано практичний досвід налаштування та тестування IDS, що є важливим етапом для побудови надійної системи інформаційної безпеки в мережевому середовищі.

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				
Змн.	Арк.	№ докум.	Підпис	Дата		47

ВИСНОВКИ

У кваліфікаційній роботі було здійснено повноцінне дослідження, розробку, впровадження та тестування підсистеми виявлення мережових вторгнень з використанням інструменту Snort. Робота охоплює як аналіз існуючих рішень IDS із відкритим кодом, так і практичну реалізацію функціонального середовища для виявлення атак, що становлять загрозу безпеці комп'ютерних мереж.

На основі аналізу літературних джерел і практичного тестування зроблено висновки, що Snort є ефективним та гнучким інструментом для побудови підсистеми мережевого моніторингу. Його архітектура дозволяє реалізовувати складні правила виявлення, комбінуючи сигнатурний та поведінковий підходи. Завдяки відкритому вихідному коду, підтримці гнучкої конфігурації, широкому набору готових сигнатур і можливості створення власних, Snort є придатним не лише для навчальних цілей, а й для реального використання у продуктивних середовищах малого та середнього масштабу.

У роботі реалізовано практичну модель підсистеми виявлення вторгнень у віртуальному середовищі на основі GNS3 та VirtualBox. Було інстальовано Snort на базі ОС Ubuntu, налаштовано мережу для перехоплення трафіку, створено набір правил, що дозволяють виявляти типові атаки, зокрема ICMP flood, підозрілу SMTP-активність, TFTP-запити тощо. Система успішно продемонструвала свою здатність до фіксації загроз, створення журналів інцидентів і генерації сповіщень.

Окремим досягненням дослідження стало вивчення та впровадження методів узагальнення сигнатур через модифікацію та інверсію параметрів правил. Застосування інструментів FuzzRule і AlertMerge дозволило провести порівняльний аналіз результатів детекції, виявити вплив спрощених правил на точність системи, а також виявити пріоритетну поведінку Snort щодо спрацювання саме узагальнених варіантів. Це дало змогу запропонувати ефективний підхід до створення правил, який знижує кількість хибнопозитивних сповіщень і підвищує загальну точність системи.

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				
Змн.	Арк.	№ докум.	Підпис	Дата		48

Разом з тим, робота виявила низку обмежень Snort, які стосуються масштабованості в умовах високонавантажених мереж, складності оптимізації великої кількості правил, а також потреби в інтеграції з іншими системами для централізованої обробки подій.

У підсумку можна стверджувати, що поставлена в роботі мета була повністю досягнута. Розроблена підсистема продемонструвала здатність до виявлення широкого спектру загроз у локальній мережі та може слугувати основою для подальшого вдосконалення. Зокрема, доцільним є розширення функціоналу за рахунок переходу до Snort 3, автоматизація оновлення сигнатур, інтеграція з системами аналітики, а також впровадження технологій штучного інтелекту для адаптивного реагування на нові типи атак.

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				
Змн.	Арк.	№ докум.	Підпис	Дата		49

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Gupta, A., & Sharma, L. S. (2020). A categorical survey of state-of-the-art intrusion detection system-Snort. *International Journal of Information and Computer Security*, 13(3/4), 337-356.
2. Риків, В. (2020). Порівняльний аналіз систем виявлення і запобігання вторгнень. Доступно за адресою: <https://openarchive.nure.ua/bitstream/document/10555/1/RYSKOV.pdf>
3. Understanding and Configuring Snort Rules. (2016). Доступно за адресою: <https://www.rapid7.com/blog/post/2016/12/09/understanding-andconfiguring-snort-rules/>
4. What is Snort?. Доступно за адресом: <https://www.fortinet.com/resources/cyberglossary/snort>
5. 10 найкращих систем виявлення вторгнень. Доступно за адресою: <https://uk.myservername.com/top-10-best-intrusion-detection-systems>
6. Roesch, M. (1999). Snort: Lightweight intrusion detection for networks (Vol. 229). Santa Clara, CA: Stanford Telecommunications Inc.
7. Norton, M., & Roelker, D. (2002). SNORT 2.0: Hi-performance multi-rule inspection engine. Columbia: Sourcefire Network Security Inc.
8. Caswell, B., Beale, J., & Baker, A. (2007). Snort IDS and IPS toolkit. New York: Syngress.
9. Tjhai, G. C., Papadaki, M., Furnell, S. M., & Clarke, N. L. (2007). Investigating the problem of IDS false alarms: An experimental study using Snort. In *Proceedings of the International Information Security Conference*.
10. Zhou, Z., Zhongwen, C., & Tiecheng, Z. (2010). The study on network intrusion detection system of Snort. In *2010 2nd International Conference, IEEE*.
11. Zhang, X., Li, C., & Zheng, W. (2004). Intrusion Prevention System Design. *The Fourth International Conference on Computer and Information Technology (CIT'04)*, 65.
12. Методи ідентифікації аномальних станів для систем виявлення вторгнень. (2019). Доступно за адресою: <https://dSPACE.nau.edu.ua/handle/NAU/38772/>

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				50
Змн.	Арк.	№ докум.	Підпис	Дата		

13. Lu, P. (2013). A lightweight intrusion detection system Snort. Silicon Valley, 4, 57-57.
14. Sagala, A. (2016). Automatic SNORT IDS rule generation based on honeypot log.
15. Kuvaiskii, D., Chakrabarti, S., & Vij, M. (2018). Snort intrusion detection system with Intel Software Guard Extension (Intel SGX).
16. Tjhai, G. C., Papadaki, M., Furnell, S. M., & Clarke, N. L. (2008). Investigating the problem of IDS false alarms: An experimental study using Snort. Proceedings of the International Conference.
17. Park, W., & Ahn, S. (2017). Performance comparison and detection analysis in Snort and Suricata environment. Wireless Personal Communications.
18. Liang, J. (2015). Based on the Snort network intrusion defense explored.
19. SNORT IDS. Доступно за адресою: <http://www.snort.org/>
20. Ubuntu OS. Доступно за адресою: <https://ubuntu.com/>

		Гайдученко Є.О			ККІКБ.КР.Б – 125 – 04 – ПЗ	Арк.
		Покотило О.А				51
Змн.	Арк.	№ докум.	Підпис	Дата		