Хмельницький національний університет
Факультет інформаційних технологій
Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

Матюха Дмитра Анатолійовича

на здобуття ступеня вищої освіти Бакалавра

Система централізованого журналювання DNS-запитів

Галузь знань 12 – Інформаційні технології

Спеціальність 125 – Кібербезпека

Освітня програма Кібербезпека

КРБКБ. 220115.22.01.09 ПЗ

Виконав студент 4 курсу, група КБ-22-1 _____ Дмитро МАТЮХ
Підпис, дата — Ініціали, прізвище

Керівник канд. тех. наук, доцент _____ Юрій КЛЬОЦ
Науковий ступінь, вчене звання — Підпис, дата — Ініціали, прізвище

Нормоконтролер д-р філософії _____ Наталія ПЕТЛЯК
Науковий ступінь, вчене звання — Підпис, дата — Ініціали, прізвище

До захисту допускаю:

Зав. кафедри кібербезпеки _____ Юрій КЛЬОЦ
Підпис, дата — Ініціали, прізвище

16 06 2026р.

Хмельницький, 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Кібербезпеки

Рівень вищої освіти Бакалавр

Галузь знань 12 – Інформаційні технології

Спеціальність 125 – Кібербезпека

Освітня програма Кібербезпека

ЗАТВЕРДЖУЮ
Завідувач кафедри кібербезпеки
Юрій КЛЬОЦ
20 січня 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Матюха Дмитра Анатолійовича

1 Тема роботи Система централізованого журналювання DNS-запитів

Керівник роботи к.т.н, доц. зав. кафедри кібербезпеки Юрій Павлович Кльоц

Затверджено наказом ректора університету від 20 січня 2026 № 7

2 Строк подання студентом кваліфікаційної роботи на кафедру _____

3 Вихідні дані до роботи створити систему збору, обробки та зберігання мережевої активності пристроїв за протоколом DNS на основі відправки логів на зовнішній сервер для здійснення їх подальшої обробки та журналювання

4 Зміст пояснювальної записки (перелік питань, які потрібно розробити) Аналіз роботи протоколу DNS. системи збору DNS-запитів від мережевих пристроїв. Реалізація прототипу та тестування системи централізованого журналювання DNS-запитів

5 Перелік графічного матеріалу (із зазначенням обов'язкових креслень) Схема роботи DNS-протоколу. Топологія проходження мережевого трафіку у мережі. Результати тестування прототипу системи

6 Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Петляк Н.С., д-р філософії, доцент кафедри кібербезпеки		

7 Дата видачі завдання 20 січня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
Вибір і затвердження теми кваліфікаційної роботи	лютий	
Ознайомлення з предметною областю	лютий	
Дослідження існуючих рішень	лютий	
Постановка задачі	березень	
Визначення загальних принципів рішення задачі	березень	
Деталізація принципів рішення задачі	квітень	
Розробка проєктних рішень	квітень	
Апробація проєктних рішень	травень	
Оформлення пояснювальної записки згідно вимог	травень	
Оформлення графічної частини	червень	
Захист КР	червень	

Студент _____ Дмитро МАТЮХ

Керівник кваліфікаційної роботи _____ Юрій КЛЬОЦ

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Система централізованого журналювання DNS-запитів».

Автор роботи: Матюх Дмитро Анатолійович

Керівник роботи: Кльоц Юрій Павлович

Пояснювальна записка: 62 с., 2 додатки, 11 рис., 1 табл., 42 джерела.

Графічна частина: 3 плакати.

Ключові слова: система моніторннгу, мережева активність, протокол DNS, Mikrotik CHR, Docker, ELK Stack, IP-адреси, віртуалізація, атаки на мережеві протоколи, агенти для збору інформації.

У кваліфікаційній роботі досліджено роботу мережевого протоколу системи доменних імен та визначено основні вектори атак, що можуть здійснюватися для порушення його роботи. Проведено дослідження для визначення актуальності теми моніторингу мережевого трафіку. Здійснено аналіз наявних рішень.

Визначено компоненти системи централізованого журналювання DNS-запитів та вивчено особливості роботи кожного з них. Розроблено архітектуру системи та налаштовано її основні компоненти для вирішення проблеми аналізу активності пристроїв в мережі Розроблено рішення для коректної обробки вхідних повідомлень від серверів та визначено логіку агрегації даних.

Реалізовано прототип системи централізованого журналювання DNS-запитів. Здійснено повноцінне тестування функціонування прототипу системи в різних умовах роботи мережі. Проведено тестування при звичайному навантаженні в умовах роботи домашньої мережі. Здійснено симуляції атак для визначення недоліків та перевірки відмовостійкості системи під навантаженням. Результатом тестування є демонстрація методу збору та аналізу логів DNS-запитів як один із ключових інструментів забезпечення безпеки мережі.

08.06.2026

ABSTRACT

Theme of the qualification work: « Centralized DNS Query Logging System».

Author of the work: Matiukh Dmytro Anatoliiovych.

Supervisor: Klots Yurii Pavlovych.

Explanatory note: 62 p., 2 appendices, 11 figures, 1 tables, 42 references.

Graphic part: 3 posters

Keywords: monitoring system, network activity, DNS protocol, Mikrotik CHR, Docker, ELK Stack, IP addresses, virtualization, network protocol attacks, information collection agents..

The qualification thesis investigates the operation of the Domain Name System network protocol and identifies the main attack vectors that can be carried out to disrupt its functioning. A study was conducted to determine the relevance of network traffic monitoring. An analysis of existing solutions was carried out.

The components of the centralized DNS query logging system were identified, and the operational features of each component were studied. The system architecture was developed, and its main components were configured to solve the problem of analyzing device activity within the network. A solution for the correct processing of incoming messages from servers was developed, and the logic for data aggregation was defined.

A prototype of the centralized DNS query logging system was implemented. Full-scale testing of the prototype's functioning was carried out under various network operating conditions. Testing was conducted under normal load conditions within a home network. Attack simulations were performed to identify shortcomings and verify the system's fault tolerance under load. The test results demonstrate the method of collecting and analyzing DNS query logs as one of the key tools for ensuring network security.

08.06.2026

ЗМІСТ

ВСТУП

В сучасних реаліях, питання цифрової безпеки стоїть на одному з головних місць, оскільки цифрові технології використовуються повсюди: починаючи з квартири чи офісу, закінчуючи криміналістичними лабораторіями. Сьогодні цифрові пристрої доступні більшості населення, зростає попит, досвід користування різними побутовими пристроями, такими як смартфони, телевізори, пристрої IoT та інші. Але зі зростанням попиту з'являється все більше загроз цифрової безпеки суспільства через витік даних компанії, що надає додаток для листування або ж просто шкідливий файл завантажений користувачем з підозрілого сайту в інтернеті [1, 2]. Для протидії таким загрозам застосовуються сучасні стратегії та тактики кібербезпеки [3].

Враховуючи вищеперелічене, непогано було б мати систему, котра може аналізувати мережу, виявляти збої, сповіщати про підозрілі дії чи файли в системі. Саме такі функції виконують системи моніторингу та контролю подій або ж SIEM (англ. Security information and event management) [4], які часто використовують алгоритми машинного навчання для виявлення аномалій у трафіку [5].

В даній роботі ми розглянемо одну з ключових складових сучасних комплексних систем моніторингу мережі, а саме систему збору логів мережевого трафіку. Завдання полягає в проєктуванні, розробці та впровадженні системи з централізованого журналювання DNS запитів. Частина системи, що збирає логи DNS-запитів, оскільки за її допомогою можна попередити кіберінцидент з можливого витоку інформації, допомогти розслідуванню, якщо кіберінцидент відбувся та надає можливості з глибокого аналізу причин, що передували інциденту з безпеки, наприклад звертання до зовнішніх, підозрілих доменів та C2-серверів, тощо. Тому важливо враховувати всі нюанси в кожному елементі моніторингу системи, щоб в подальшому мати змогу запобігти новим загрозам, що постійно виникають у нашому житті.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ ПРО СКЛАДОВІ СИСТЕМИ ЛОГУВАННЯ МЕРЕЖЕВИХ ЗАПИТІВ

1.1 Аналіз ролі DNS-протоколу, можливі загрози та вектори атак

Основною функцією протоколу DNS (англ. Domain Name System) за визначенням є співставлення значень адрес зрозумілих людині до цифрових значень IP-адреси, за якими відбувається подальша адресація пакетів з необхідною інформацією, до прикладу співставлення домену google.com до логічної цифрової адреси 142.250.130.100 [6, 7]. Ці принципи є базовими для функціонування сучасних комп'ютерних мереж [8, 9].

Також за допомогою протоколу DNS може відбуватися перевірка автентичності ресурсів, до яких здійснюється доступ користувачем, тобто співставлення коректної пари домену та його адреси для подальшої коректної адресації до ресурсу. Але на даному етапі можуть з'являтися різні вектори атак, такі як підміна DNS відповідей (DNS Spoofing) або ж несанкціонована заміна таблиці DNS доменів в кеші проміжного резолвера (DNS Cache Poisoning), загроза яких полягає в тому, що клієнт помилково вважає, що звертається до потрібного сервера, але насправді надсилає запити до зловмисного C2 домену, внаслідок чого можлива втрата конфіденційної інформації [10, 11].

Основна схема роботи DNS-протоколу виглядає наступним чином: користувач бажає відвідати інтернет-ресурс, робить запит зі свого браузера до конкретного ресурсу, спочатку перевіряється локальний кеш пристрою на наявність IP-адреси запитуваного ресурсу, якщо ж не знаходить цієї інформації у кеші, тоді цей запит відправляється до пристрою, котрий вказаний в налаштуваннях мережі як основний DNS-сервер. Далі, цей пристрій також перевіряє локальний кеш на наявність необхідної пари Домен – IP-адреса. Якщо ж тут також відсутня інформація про ресурс, він відправляє запит до DNS-резолвера інтернет провайдера, де провайдер зазвичай зберігає інформацію про популярні ресурси, щоб не робити запит кожного разу до інших серверів доменних імен. Якщо ж інформації і тут немає, тоді запит передається до

серверів доменної зони, котрі обслуговують перший рівень домену, наприклад доменна зона .com або .ua, після чого, ці сервери передають запит відповідно до своїх записів до самого авторитетного сервера, тобто сервера на котрому обслуговується ресурс. Після того, як IP-адреса запитуваного сайту відома, вона повертається назад по всьому ланцюжку до пристрою, з якого здійснювався запит та записується у локальний кеш, щоб не повторювати знову пошук адреси.

Після цього вже встановлюється з'єднання між пристроєм та сервером і здійснюється передача інформації, наприклад: вебсторінка, файли, картинки.

Важливим моментом є забезпечення надійності проходження DNS-трафіку, оскільки, як вже було зазначено, на протокол DNS можуть здійснюватися атаки та існують ризики для безпеки та конфіденційності при його використанні.

Загроза безпеці прямо корелює із загрозою конфіденційності мережі чи окремого користувача. Одна з основних загроз конфіденційності є можливість витоку DNS-трафіку. Оскільки з DNS-трафіку можна безпосередньо і точно визначити до яких доменів звертався користувач та які ресурси він відвідував або ж намагався відвідати. З цього випливає загроза можливого витоку DNS-трафіку, оскільки використання неправильних налаштувань мережевого обладнання, використання ненадійного VPN або ж наявність на пристрої шкідливого програмного забезпечення, яке змінює налаштування DNS на цільових пристроях. Зазначені приклади несуть в собі загрозу, що DNS-трафік буде проходити не через авторизований та надійний сервер, а через, наприклад, C2-сервер ботнет мережі або ж інші неавторизовані та ненадійні сервери DNS, які не забезпечують належної безпеки конфіденційності трафіку. Як наслідок, може відбутися компрометація окремих користувачів, розкриття відвідуваних ресурсів та розкриття їхнього місцезнаходження базуючись на їхній IP-адресі.

Доступ зловмисників до цієї інформації може завдати критичні ризики фізичної безпеки особи у випадку розкриття відвідуваних ресурсів або ж її геолокації. Саме для того, щоб зменшити ризик витоку та зменшити ймовірність розкриття конфіденційної інформації, потрібно запроваджувати системи

моніторингу мережі, щоб відслідковувати та блокувати доступ до зловмисних або неавторизованих серверів.

1.2 Огляд інструментів для реалізації системи з моніторингу мережевого трафіку

Для реалізації системи моніторингу мережевого трафіку, необхідно спочатку переглянути наявні інструменти, за допомогою яких можна виконати завдання кваліфікаційної роботи.

Першою програмою для розгляду буде Wireshark. Wireshark являє собою безкоштовний мережевий сніфер, який може використовуватися як програма для збору пакетів мережевого трафіку [12, 13]. Інтерфейс програми Wireshark під час захоплення трафіку наведено на рис. 1.1.

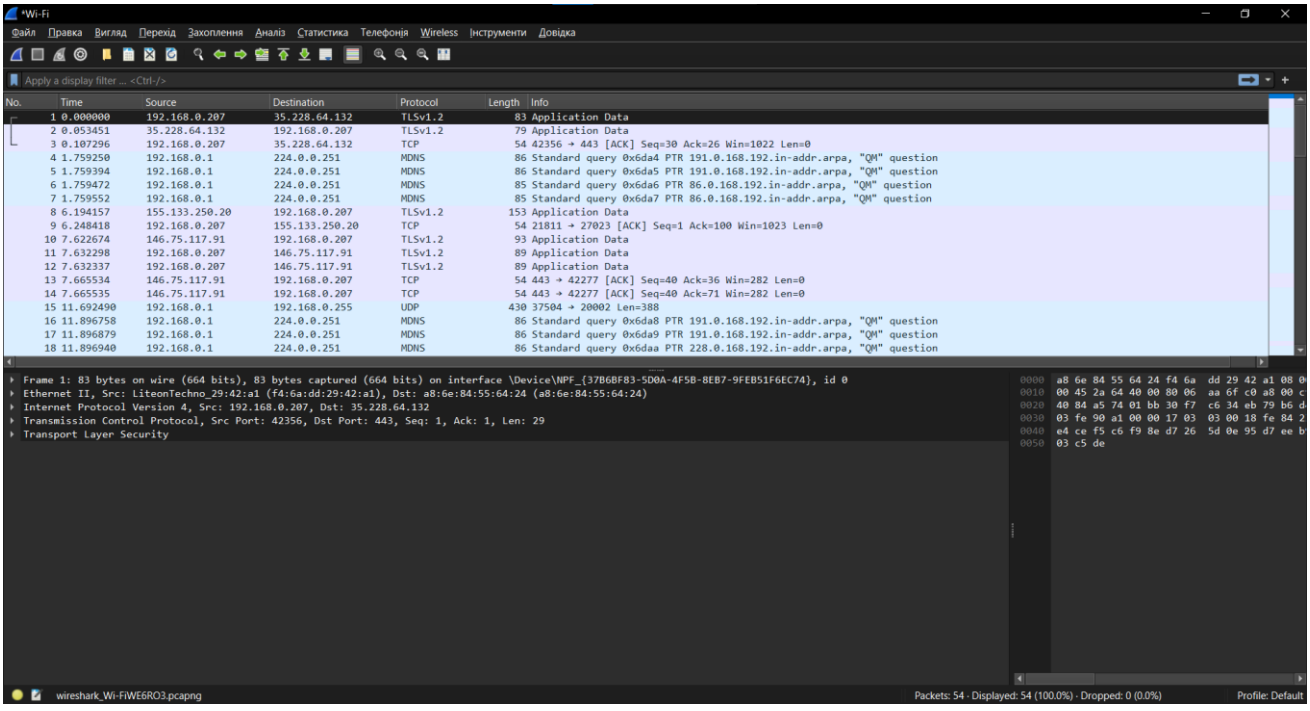


Рисунок 1.1 – Інтерфейс програми Wireshark у процесі захоплення трафіку

Також ця програма дає можливість детально переглядати вміст кожного пакету на кожному етапі еталонної моделі OSI. У ньому можна вибрати

конкретний мережевий інтерфейс пристрою для перехоплення та моніторингу трафіку, що дозволяє гнучко оперувати наявними каналами походження пакетів. З плюсів даної програми, що вона ідеально підходить для захоплення, фільтрування та детального аналізу недавніх пакетів.

Але хоч Wireshark - це ідеальний сніфер, у цій програмі проблематично здійснювати пошук чи аналіз пакетів за довготривалий проміжок часу, наприклад за місяць, квартал чи рік. Тому що під час аналізу великих обсягів захопленого трафіку, Wireshark часто перестає працювати належним чином, через що можуть виникати збої як самої програми, також перевантаження Wireshark може вплинути на роботу самої системи моніторингу, де встановлена сама програма. Найгірше, що може відбутися у випадку такого перевантаження, це ризик втрати важливих файлів захоплення, що може спричинити подальшу неможливість відстеження або ж розслідування інциденту безпеки.

ELK Stack - це програмний стек для захоплення, зберігання, аналізу та візуалізації мережевого трафіку. ELK Stack складається з трьох ключових програм: ElasticSearch, Logstash та Kibana. Кожна з програм виконує свою окрему функцію, але разом вони поєднуються у потужний інструмент для моніторингу мережевого трафіку та централізованого збору логів. Детальніше про кожну частину стеку [14].

Logstash виступає інструментом з відкритим вихідним кодом, головне завдання якого полягає в агрегації трафіку з різноманітних джерел. Процес обробки включає гнучку фільтрацію вхідних даних та їх подальшу трансформацію в уніфікований JSON-формат. Завдяки розширеній системі конфігурацій, цей компонент здатний ефективно розпізнавати необхідні поля, проводити агрегацію для зв'язування подій (наприклад, співставлення IP-адреси з конкретним джерелом) та відсікати надлишкову інформацію з мережевих пакетів, мінімізуючи навантаження на кінцеву базу даних [15].

Elasticsearch відіграє роль високопродуктивного централізованого сховища, що відповідає за глибоку індексацію та миттєвий повнотекстовий пошук серед величезних масивів неструктурованої інформації. Написана на мові

Java, ця система активно взаємодіє з іншими вузлами через REST API. Її архітектурна перевага полягає у використанні інвертованого індексу: замість класичного перебору рядків, механізм одразу вказує на конкретний документ, де міститься шуканий параметр. Саме такий підхід робить Elasticsearch ідеальним рішенням для швидкого аналізу логів [16].

Kibana слугує зручним вебінтерфейсом, який має пряму інтеграцію з Elasticsearch через REST API. Вона не лише дозволяє керувати базою даних через браузер, але й надає широкі можливості для створення кастомних візуалізацій, дашбордів та моніторингу цілісності системи. Важливою функцією є гнучке налаштування політик доступу (Role-Based Access Control) та створення автоматичних сповіщень (Alerts) — наприклад, при фіксації підозріло високої кількості запитів до сумнівних серверів, що є критично важливим для оперативного реагування на кіберінциденти [17].

Додатково, в Kibana можна створювати групу умов, при виникненні яких будуть відображатися сповіщення (Alerts), наприклад, коли пристрій надсилає запит до підозрілого сервера або ж надсилає багато запитів в секунду. Ці механізми допомагають ефективно здійснювати контроль над безпекою та коректною роботою мережі.

Docker – платформа для реалізації розгортання процесів в ізольованих контейнерах, що дозволяє гнучко керувати, масштабувати та проводити аналіз компонентів системи. Використання Docker як середовища для побудови централізованої бази даних мережевого трафіку забезпечує мінімальне використання ресурсів хостової машини, що робить саме такий підхід більш бюджетним у порівнянні з розгортанням повноцінних віртуальних машин для кожного окремого процесу [18].

Контейнеризація забезпечує сегментацію системи, тобто, якщо один її компонент вийде з ладу це критично не вплине на роботу інших процесів. Також, в разі проблеми, збою чи несанкціонованого зараження компоненту системи на платформі Docker, це вплине лише на ізольований контейнер і не призведе до критичного стану хостової машини.

Splunk – повноцінна комерційна система для аналізу даних, на якій будуються SIEM-рішення, що дозволяє ефективно збирати, аналізувати, здійснювати пошук та реагувати на інциденти. В основному використовується у великих компаніях, оскільки дозволяє швидко та надійно обробляти великі обсяги інформації, що є критично для корпорацій. Splunk - це комерційний продукт, без відкритого вихідного коду, розповсюдження якого здійснюється за допомогою купівлі ліцензії на користування продуктом [19].

Оскільки ліцензія на Splunk потребує значних фінансових витрат, цей варіант було вирішено не розглядати в рамках реалізації завдання дипломної роботи.

1.3 Роль централізованої системи збору логів у забезпеченні безпеки мережі та побудова прототипу системи

Отримання та зберігання відомостей про події, що відбувалися в мережі є важливими кроками для аналізу функціонального стану мережі. Система зі збору та зберігання повинна відповідати декільком ключовим вимогам [20, 21]:

– великий обсяг пам’яті для зберігання логів або можливість розширення обсягу пам’яті;

– простота масштабування, можливість збільшення кількості джерел трафіку під потреби з розширення мережі;

– неможливість несанкціонованого порушення цілісності збереженої інформації.

Розглянемо можливу реалізацію системи збору та зберігання мережевих логів на маршрутизаторі.

Дані про мережевий трафік, в такому випадку будуть зберігатися на маршрутизаторі, через який здійснюється вихід у загальну мережу. Але такий підхід має декілька нюансів, які варто розглянути.

По-перше, це обмеженість ресурсів пам’яті роутера. Більшість роутерів

має невелику кількість ефективного місця для зберігання файлів логів у своїй пам’яті. Якщо це мережа підприємства, де одночасно відбувається безліч подій мережевої активності, тоді пам’ять роутера закінчиться дуже швидко. Один з шляхів подолання цієї проблеми є розширення пам’яті роутера за рахунок встановлення зовнішнього накопичувача або SD-картки. Але такий варіант реалізації не є ефективним, оскільки він не передбачає можливість безпроблемного масштабування та відмовостійкості.

Також, існує ймовірність що система може вийти з ладу через перевантаження, внаслідок чого, важливі дані про події в мережі можуть бути незафіксовані або втрачені. Або ж зловмисник може спробувати видалити дані про свою мережеву активність з маршрутизатора і тоді важливі дані також будуть втрачені.

Пошук та аналіз інформації у базі даних. Проведення аналітики зібраних логів може бути проблематичним через відсутність зручної системи пошуку в файловій системі маршрутизатора.

Враховуючи зазначені фактори, збір та зберігання даних у самому маршрутизаторі не є доцільним, навіть для звичайної домашньої мережі.

Саме тому, для реалізації системи та вирішення зазначених проблем було запропоновано використання централізованої системи на базі ELK Stack.

Основна відмінність запропонованої системи в тому, що вона є централізованою та окремою від основної мережі. Це означає, що головний маршрутизатор, крізь який проходить трафік, формує та відправляє логи про мережеві події на сервер обробки та зберігання інформації, де встановлено систему ELK Stack, розгорнуту на базі середовища Docker. Використання саме цього способу запобігає виникненню проблеми з переповненням пам’яті маршрутизатора або ж перевантаження його системи, оскільки обов’язки з збору, обробки та зберігання інформації делеговані на окремий сервер.

На сервері обробки буде реалізована система у середовищі віртуального контейнера Docker, що дозволить розділити зберігання даних від функціонування основного хостового сервера обробки. Також, при

масштабуванні або перенесенні системи зберігання DNS-запитів можна перенести всю конфігурацію системи без потреби заново налаштовувати та швидко забезпечити розгортання та функціонування системи на новому сервері.

Кожен компонент ELK Stack виконує свою окрему функцію та вирішує кожну з визначених проблем. Розіб'ємо схему роботи системи на етапи від джерела трафіку до подальшого аналізування зібраних логів. Спочатку, пристрої відправляють запити до інтернет-ресурсів, тим самим виступаючи як генератори трафіку. Далі, вся інформація проходить через маршрутизатор на базі Mikrotik, за операційною системою RouterOS, який володіє функцією генерування та відправки логів на віддалений сервер.

Logstash виконує функцію обробки та фільтрування логів перед їх подальшою передачею до бази даних Elasticsearch. Функція фільтрування дозволяє зменшити розмір конкретного логу шляхом відкидання непотрібної інформації та приведення кінцевого логу до єдиного структурованого вигляду.

Elasticsearch – це функціональне ядро системи, що повністю вирішує проблему з пам'яттю, індексацією та пошуком необхідної інформації у великих обсягах даних. Враховуючи, що база даних розташована на окремому сервері, це означає, можливість збільшення обчислювальних можливостей та необмежений простір для зберігання великої кількості мережевих логів. Також швидкість роботи бази даних повністю вирішує проблему з пошукум потрібної інформації. Навіть у випадку компрометації основного вузла на базі Mikrotik, для бази даних з логами мережевого трафіку ризик порушення цілісності мінімальний.

Kibana – виступає в ролі вебінтерфейсу для бази даних Elasticsearch. За допомогою цього компонента системи можна здійснювати пошук у зручному користувацькому інтерфейсі, керувати правами доступу та створювати якісну аналітику для спрощення моніторингу подій у мережі.

Отже, зваживши перелічену інформацію та відповідні ключові критерії, для реалізації завдання дипломної роботи було вирішено вибрати ELK Stack. У другому розділі буде зазначена практична реалізація та налаштування ключових компонентів системи централізованого збору та зберігання DNS-запитів.

1.4 Висновки до розділу

У першому розділі було проаналізовано роль DNS-протоколу у забезпеченні належного функціонування будь-якої сучасної мережі. Визначено основні типи та вектори атак на даний протокол, та загрозу, яку вони в собі несуть. Додатково, було визначено якими мають бути вимоги для системи централізованого збору DNS-запитів. Проведено аналіз важливості такої системи для забезпечення безпеки мережі підприємства чи звичайної домашньої мережі. Розглянуто питання актуальності використання такої мережі.

Також, велика частина розділу відведена під вибір основного ядра системи. Було зважено декілька важливих факторів, такі як:

- простота масштабування;
- доступність;
- неможливість несанкціонованої зміни зібраних даних;
- гнучкість налаштувань системи.

За вказаними критеріями кращими програмними рішеннями для реалізації завдання кваліфікаційної роботи було визначено ELK Stack, Mikrotik CHR та Docker. Обґрунтовано вибір саме такого програмного забезпечення тим, що кожен елемент системи вирішує свою проблему з вищезазначених. Наприклад, Elasticsearch та Kibana повністю вирішують проблему з доступністю. Тому що поєднання цих програм є база даних Elasticsearch та вебінтерфейс Kibana. Що без проблем інтегровані у даному програмному стеку. Вибір контейнерного середовища Docker зумовлено тим, що він повністю вирішує проблему масштабованості. Тому що до нього можна під’єднати велику кількість різних агентів зі збору інформації. Це дозволяє легко масштабувати мережу, без необхідності зміни конфігурації основної системи.

Обґрунтування вибору віртуального маршрутизатора Mikrotik тим, що він може без проблем відправляти логи про перехоплений трафік на зовнішню адресу. Що у побудові системи зі збору та зберігання є однією з основних умов. Також, наявність такого функціоналу вирішує проблему з неможливістю

несанкціонованої зміни зібраних даних. Це вирішується завдяки тому, що якраз маршрутизатор відправляє логи одразу на зовнішню адресу. І навіть у випадку компрометації самого роутера, зловмисник може змінити налаштування в ньому або ж видалити дані про свою активність. У випадку відправки логів на сторонній сервер, зловмисник вже не може просто так стерти свою активність, оскільки це вимагатиме додаткового зламу і системи зберігання логів. А при забезпеченні достатнього рівня захищеності такої системи, ризик компрометації і втрати цілісності збережених даних є доволі низьким.

Оскільки, вибрані рішення повністю вирішують поставлені вимоги до системи саме час переходити до налаштування цієї системи.

2. ПРОЄКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ ЦЕНТРАЛІЗОВАНОГО ЖУРНАЛЮВАННЯ DNS-ЗАПИТІВ

2.1 Розробка схеми топології зв’язку між ключовими вузлами системи

Потрібно визначити основні вузли системи, через які буде проходити інформація та встановити роль кожного вузла.

Схема проходження DNS-трафіку від початкового запиту клієнтського хосту до зберігання у базі даних та подальшій візуалізації буде розроблена згідно у попередньому розділі вибраних програмних рішень.

Для функціонування системи потрібно джерело трафіку. В ролі такого можуть виступати будь-які клієнтські хости, що під’єднуються до мережі та генерують DNS-запити.

Далі нам необхідний вузол, котрий буде виступати в ролі шлюзу та локального DNS-резолвера для усієї мережі та, в подальшому, обробляти та пересилати логи про DNS-трафік на сервер з ELK Stack. В ролі DNS-резолвера буде виступати маршрутизатор Mikrotik, що розгорнутий на віртуальній машині за допомогою гіпервізора другого типу VirtualBox [22]. Вибір програмного рішення віртуалізації від VirtualBox зумовлений для розробки та тестування функцій маршрутизатора Mikrotik, без необхідності придбання апаратного продукту. Mikrotik надає безкоштовне рішення для віртуального розгортання маршрутизатора за допомогою рішення MikroTik Cloud Hosted Router (CHR) [23]. Ця технологія розповсюджується за допомогою ліцензії. Типи ліцензування для Mikrotik CHR представлені у таблиці 2.1.

Таблиця 2.1 Типи ліцензування для Mikrotik CHR

Назва ліцензії	Обмеження швидкості	Наявність функцій
free	1 Mbps	Всі
p1 (perpetual-1)	1 Gbps	Всі
p10 (perpetual-10)	10 Gbps	Всі
p-unlimited	Необмежено	Всі

Для реалізації задачі дипломної роботи, в тестових цілях, вистачить типу ліцензії free, оскільки пропускна здатність у 1 Mbps буде цілком достатньою, для проєктування демонстраційного прототипу, оскільки DNS створює мінімальне навантаження на мережу.

На даному етапі ми маємо джерело трафіку та основний DNS-сервер, крізь який будуть проходити усі запити мережі. Наступною ланкою буде проходження сформованих Syslog-повідомлень від Mikrotik до середовища контейнеризації Docker, у якому розгорнуто ELK Stack [24, 25].

У середовищі VirtualBox для цього налаштовано декілька мережевих інтерфейсів. Перший інтерфейс Bridge, що дає можливість DNS-запитам проходити через віртуальне середовище до MikroTIK, та забезпечує безперебійне отримання DNS-відповідей із зовнішньої мережі. Також створено та налаштовано віртуальний інтерфейс для внутрішньої мережі, для безпечного та надійного пересилання Syslog-повідомлень до середовища контейнеризації Docker.

Після входження у віртуальну мережу середовища Docker, Syslog-повідомлення потрапляє спочатку у контейнер Logstash, де буде відбуватися фільтрація, обробка та перетворення Syslog-повідомлень у JSON-формат. Також, на цьому етапі буде відкидатися непотрібна частина, щоб зменшити навантаження на базу даних. Після проходження через фільтр Logstash, вже структуровані логи потрапляють до бази даних Elasticsearch, де вони індексуються відповідно до налаштованих правил та зберігаються у системі для їх подальшого перегляду через вебінтерфейс Kibana. У вебінтерфейсі здійснюється налаштування відображення логів, відповідно до здійсненої індексації у базі даних. У Kibana, за допомогою налаштування в секції Data View, може бути додатково налаштовано відображення одразу кількох типів логів з логічним розділенням на потоки кожних логів.

2.2 Аналіз та конфігурація мережевого обладнання

Ключовим мережевим обладнанням в топології виступає MikroTik CHR. Наступним кроком буде розглянуто мережеві налаштування та формування Syslog-повідомлень у сховищі маршрутизатора.

Спочатку необхідно створити та налаштувати віртуальну машину у VirtualBox. Для цього був завантажений образ MikroTik CHR у форматі VDI. Після чого у програмному середовищі VirtualBox створюємо віртуальну машину з образу, відповідно до офіційної документації від MikroTik.

Первинне встановлення параметрів оперативної пам'яті та дискового простору для операційної системи RouterOS, якими вона зможе оперувати. Відповідно до документації, рекомендований обсяг пам’яті становить 128 мегабайт дискового простору та 256 мегабайт оперативної пам’яті. Враховуючи, що для тестового варіанту пропускна здатність інтерфейсів має обмеження в 1 мегабіт на секунду, що не буде створювати надмірного навантаження на віртуальну систему, вистачить 128 мегабайт місця на віртуальному диску та 512 мегабайт оперативної пам’яті. Наступним кроком буде налаштування мережевих інтерфейсів для роутера у середовищі VirtualBox. Перший інтерфейс буде мати тип Міст. Він буде призначений для того, щоб віртуальний роутер мав вихід в інтернет та міг надсилати запити DNS та отримувати на них відповіді. Додатково, для подальших тестів налаштуємо внутрішній інтерфейс для роутера. Для того, щоб для подальшого тестування, можна було наприклад створити віртуальні машини на базі різних операційних систем і під'єднати їх до цієї внутрішньої мережі Mikrotik. Іншими словами, тестування системи у повністю ізольованому середовищі від фізичної локальної мережі.

Після того, як попередні налаштування у VirtualBox були завершені, приступаємо до наступного кроку, а саме встановлення та налаштування віртуального роутера. Є декілька способів налаштування обладнання від Mikrotik. Перший спосіб це використання командного рядка. Це точний та швидкий спосіб, але він вимагає глибокого знання та розуміння синтаксису

консольного середовища Mikrotik, такі як змінні та параметри. Також, цей спосіб не є достатньо зручним для швидкого перегляду налаштувань та майбутнього потоку логів DNS-трафіку. Другий спосіб це графічний інтерфейс офіційного програмного забезпечення від Mikrotik – утиліта Winbox [26]. Вона, як і будь-який інший графічний аналог командного рядка є більш зручна в користуванні та налаштуванні окремих параметрів маршрутизатора. Також, вона підтримує режим перегляду декількох вкладок одразу. Це дозволяє швидко налагоджувати декілька параметрів одразу, використовуючи звичайне переключення між вікнами програми. Додатково, функція використання декількох вікон, дозволяє і налаштовувати додаткові параметри і одразу переглядати вплив застосованих налаштувань на поведінку мережі у вікні відображення системних логів або логів мережевої активності. Тому, врахувавши переваги графічного інтерфейсу перед командним рядком, вирішено використовувати саме утиліту Winbox.

Для початку налаштовуємо мережеві інтерфейси, через які буде здійснюватися подальше підключення та проходження трафіку. Таким чином було створено два мережеві інтерфейси: ether1 та ether2. Інтерфейс ether1 буде виконувати функцію WAN, тобто виходу в зовнішню мережу та надсилання та отримання DNS-трафіку. Тоді як інтерфейс ether2 налаштовано для внутрішньої мережі віртуального роутера або ж LAN. Це зроблено для того, якщо з'явиться потреба підключити деякі віртуальні машини напряму до маршрутизатора.

Додано правила Redirect [27]. Це зроблено для того, щоб маршрутизатор Mikrotik здійснював примусове перенаправлення усіх пакетів за протоколами UDP та TCP, що надходять на 53 порт роутера. Враховуючи специфіку роботи DNS-протоколу, що він працює за допомогою обох протоколів передачі інформації, це було виконано, щоб не втратити жодного пакету. До того ж, значення для порта конкретно 53, оскільки це є стандартний порт на якому працює DNS-протокол. Також, в правилах перенаправлення додано прапорець для логування кожного запиту і перенаправлення. Це зроблено із ціллю забезпечення видимості повної картини мережевої активності пристроїв.

Навіть якщо пристрій вкаже у своїх налаштуваннях сторонній DNS-сервер,

завдяки цим правилам, роутер Mikrotik перенаправить ці запити на свій внутрішній DNS. Це дозволить зафіксувати кожен мережевий DNS-запит від пристроїв для подальшого аналізу. А прапорець log=yes потрібен для того, щоб, в подальшому, мати можливість відслідковувати який саме пристрій намагається обійти систему моніторингу. Це налаштування забезпечує, що будь-яка нетипова мережева активність не залишиться непоміченою та все буде зафіксовано.

Оскільки, це віртуальний роутер, що знаходиться всередині іншої машини, в нашому випадку ноутбука, мережеві налаштування він буде отримувати від зовнішнього роутера. Для цього налаштовуємо DHCP-Client на Mikrotik, щоб він міг автоматично отримати мережеві налаштування. Налаштування проводиться для інтерфейсу ether1, оскільки він використовується Mikrotik'ом для виходу в зовнішню мережу. Наступним кроком буде налаштування DNS. Єдине налаштування, яке застосовано на даному етапі, це прописування основним резолвером DNS-сервери компанії Google, тобто 8.8.8.8. Додатково можна прописати і інші сервери, але для тестування прототипу вистачить і одного.

Оскільки, маршрутизатор Mikrotik виступає в ролі головного вузла перенаправлення Syslog-івентів, було налаштовано декілька правил NAT. Правила налаштовано, щоб DNS-трафік, якщо пристрої надсилають DNS-запити, то роутер їх перехоплює та спочатку надсилає їх на локальний резолвер. Ця дія дозволяє ефективно збирати мережеві події, навіть якщо деякі пристрої в своїх налаштуваннях вказали інші DNS-сервери. Тепер, коли основні мережеві налаштування виконані, потрібно налаштувати логіку логування та перенаправлення Syslog на зовнішню адресу.

Для цього у вкладці Logging, було створено два правила з назвою dns. Обидва правила налаштовані збирати події з топіками dns, що означає збір подій пов'язаних конкретно із DNS-трафіком. Також, завдяки налаштуванню правил, збираються не тільки чисті логи про запит або відповідь, а й технічна інформація. Така як наприклад дамп пам'яті в шістнадцятковому форматі або ж детальні дані кожного DNS-пакета. Єдина відмінність між ними, це дія, яку вони здійснюватимуть із цими логами. Одне з правил буде основним для проєкту,

воно використовує дію Remote, яка перенаправляє логи на зовнішню адресу.

В другому правилі вказано дію Memory. Це зроблено для зручного налагодження та перегляду чи коректно працює система логування. Також, правило запису в оперативну пам’ять роутера, дозволяє ефективно налагоджувати та виявляти аномалії в конфігурації. Як, до прикладу, якщо IP-адреса була некоректно записана до бази даних, тоді за допомогою цього правила, можна знайти цю подію в логах в самому інтерфейсі Mikrotik. Тобто ще до того, як інформація про події була відправлена до зовнішньої системи обробки.

Також, це правило дозволяє чітко побачити структуру DNS-запитів та відповідей. Це надає можливість, в подальшому, гнучко налаштувати Grok-фільтри, для Logstash.

Далі, налаштовано дію Remote, яка буде здійснювати пересилання логів на зовнішню адресу. В параметрах вказано адресу ноутбука 192.168.0.207, на якому розгорнутий маршрутизатор і де буде розгорнута база даних. Також, враховувавши, що Logstash працює на порту 514, в налаштуваннях дії Remote на Mikrotik, було вказано саме цей порт для пересилання. Тому що без правильно вказаного порта, на який буде здійснюватися пересилання логів, система обробки просто не побачить надіслані пакети з інформацією [28].

На даному етапі конфігурація маршрутизатора Mikrotik є повністю завершеною, і він успішно виконує функцію перехоплення та пересилання мережевих логів. Таким чином, базову інфраструктуру підготовлено, тому можна переходити до наступної ключової ланки системи — налаштування середовища Docker та розгортання контейнерів, у яких працюватиме програмний стек ELK Stack.

2.3 Налаштування компонентів системи централізованого зберігання DNS-запитів

Наступним кроком буде налаштування та забезпечення безперебійної роботи компонентів системи обробки та зберігання DNS-запитів.

Так як прототип системи розгортається на ноутбуці з ОС Windows, було завантажено образ програми Docker конкретно під цю операційну систему. Після встановлення потрібно налаштувати програму, щоб вона знала як працювати і які програми та параметри встановлювати для контейнерів. Для цього створено файл конфігурації docker-compose.yml [29].

Налаштування відбувається за допомогою формату YAML, що на даний час широко використовується для написання конфігураційних файлів для різних потреб. Також, цей формат має відмінності від звичайних конфігураційних форматів. Наприклад, форматі YAML не використовується табуляція, а лише окремі пробіли. Тут немає звичних фігурних дужок та крапки з комою в кінці кожного рядка. Цей формат дозволяє вільно читати конфігурацію як людині, так і комп'ютеру, що дозволяє зручно налаштовувати компоненти системи [30].

У цьому файлі описано створення окремих контейнерів та їхні параметри для кожного з компонентів ELK Stack. Для Elasticsearch вказана назва контейнеру elasticsearch, щоб інші контейнери могли до нього звертатися не за адресою, а за ім'ям контейнера у внутрішньому середовищі Docker. Також вказано, що він доступний на порту 9200 локального хоста, та виділена пам'ять для цього контейнера становить 1 гігабайт. Для контейнера Logstash вказано функціональні порти 514 та 514 UDP. Також вказано 2 гігабайти пам'яті, що виділена під цей окремий контейнер. Та налаштовано, що він звертатиметься до бази даних Elasticsearch за ім'ям контейнера elasticsearch. Для вебінтерфейсу Kibana, налаштовано контейнер з класичними портами 5601 та обмеженням пам'яті у 1 гігабайт. Аналогічно налаштовано зв'язок з базою даних за ім'ям контейнера elasticsearch.

Наступним кроком нам потрібно лише налаштувати Logstash і тоді

система буде повністю готова до тестування. Конфігурація фільтра Logstash є дуже важливим етапом для забезпечення роботи усієї системи. Оскільки він слугує фільтром між Syslog-івентами від Mikrotik та даними, що потраплятимуть до бази даних Elasticsearch. Раніше, в налаштуваннях маршрутизатора, ми вказували відправляти логи на адресу ноутбука та на порт 514, як TCP, так і UDP-пакети. На даному етапі, у файлі конфігурації Logstash ми вказуємо аналогічний порт для запуску слухача. Він пасивно слухає зазначений порт і якщо йому надсилають дані, у нашому випадку Syslog від маршрутизатора Mikrotik, Logstash пересилає ці дані далі по конвеєру обробки. Наступним кроком прописано виокремлення з сирого логу параметра id. Залежно від того, який це саме тип логу. Оскільки Mikrotik може відправляти як чисті DNS-івенти, так і логи, що переповнені технічною інформацією, наприклад номер комірки пам’яті для процесу. В роутері налаштовано пересилання лише чистих логів, тобто налаштування topics: dns, !packet. Це дозволяє відсікти зайву технічну частину, що дозволяє зменшити навантаження на систему та оптимізувати корисне навантаження для зберігання. Параметр id у різних типах логів виглядає по різному. У топіках dns він виглядає як #123, коли у топіках packet, він виглядає як id:123. В цьому блоці ми виокремлюємо з повідомлення ідентифікатор та зберігаємо його у новий параметр query_id. Це зроблено для того, щоб підвести всі логи під один стандартний вигляд. Це дозволить полегшити подальший пошук та аналіз логів.

Наступним у файлі конфігурації йде блок grok [31]. Він виконує функцію розбору вхідного тексту на параметри, які потім можна буде використовувати для подальшого аналізу інформації або ж агрегації даних. Цей блок розкладає текст на наступні змінні:

– mikrotik_time змінна, що записує час логування конкретно в самому маршрутизаторі. Це вказано для того, щоб мати змогу точно ідентифікувати подію. Без цього параметра, події будуть записуватися в базу даних лише з вказанням часу, коли вони надійшли до блоку обробки Logstash;

– clean_message змінна, в яку записується надіслане повідомлення від

Mikrotik. Зроблено це тому, щоб мати можливість аналізувати конкретне повідомлення;

– query_id ідентифікатор події, який присвоюється маршрутизатором до кожного запиту та відповіді;

– client_port порт, з якого клієнтський пристрій надіслав запит;

– domain домен, до якого клієнт надсилає запит, наприклад google.com;

– query_type тип DNS-запиту, такий як A, AAAA, TXT, MX;

– dns_class класифікація запиту, в більшості випадків це клас IN. Запит для доступу до зовнішньої мережі;

– resolved_ip визначена IP-адреса запитуваного домену;

– dns_error_code код помилки, при виникненні такої під час запиту або відповіді до сервера. Приклад NXDOMAIN, тобто неіснуючий домен;

– dns_error_message повідомлення з помилкою DNS;

– packet_action дія, що зроблена над пакетом. Тобто, фіксація того чи його надіслали або отримали;

– remote_server_ip IP-адреса зовнішнього DNS-сервера, до якого звертався Mikrotik;

– remote_server_port порт віддаленого сервера, з якого надійшла відповідь на запит.

Це буде використовуватися для подальшого фільтрування логів за IP-адресою, доменами або ж типами DNS-записів. Наприклад для того, щоб побудувати візуалізацію. Це дозволить побачити до яких доменів зверталися конкретні адреси. Після успішного розбору тексту на зручні патерни, в кінець структурованого логу додається тег dns_parsed.

Наступним кроком буде налаштування блоку агрегації [32]. Цей блок дозволить нам в подальшому будувати якісну візуалізацію та ефективно аналізувати захоплені логи. Агрегація виконує функцію співставлення різних івентів. На початку конфігурації ми прописали відділення ідентифікатора від тексту пакету. Саме ось цей ідентифікатор і дає можливість здійснювати агрегацію. Оскільки, наприклад адреса пристрою, що запитує домен,

знаходиться лише у першому запиті. При отриманні відповіді на цей запит адреса пристрою не зберігається. Тобто формат повідомлення виглядає так: запит від 192.168.0.207, відповідь на запит: Домен google.com, адреса домену 142.250.130.101, тип запису A. Для того, щоб зв'язати, що це відповідь на запит від конкретного пристрою використовується query_id. Це працює наступним чином: пристрій надсилає запит, роутер формує лог і створює ідентифікатор, наприклад 12. Для DNS-відповідей, що приходять наступними він також маркує їх ідентифікатором 12. Через перший фільтр у Logstash ми витягуємо цей ідентифікатор основного повідомлення. Після чого ідентифікатор 12 тимчасово асоціюється з адресою пристрою 192.168.0.207 на 10 секунд. Протягом яких у відповіді які мають такий самий ідентифікатор приписується адреса початкового пристрою та лог записується у базу даних. Після того, як спливає таймер у 10 секунд, асоціація між ідентифікатором та адресою видаляється, що дозволяє економити обсяг оперативної пам’яті.

Агрегація здійснюється також для поля client_port. Це зроблено для того, щоб мати додаткову можливість в подальшому коректно зв’язувати події між собою для детального аналізу мережевої активності.

Пройшовши через блок агрегації, ми маємо структуровані та зв’язані логи, за якими можна відслідкувати як відбувались події.

Після цього Logstash відсилає готові структуровані JSON-документи з індексом формату dns-logs-%{+YYYY.MM.dd} у базу даних Elasticsearch на подальше зберігання. В базі даних вони автоматично індексуються та структуруються відповідно до логіки роботи Elasticsearch. Кожен індекс містить в собі документи, що відповідають записам про кожен лог в системі.

Додатково було вирішено додати автоматичну перевірку доменів у Logstash. Для цього додано ще одну перевірку, під час якої витягується назва домену, наприклад google.com, співставляється зі локальним словником шкідливих доменів. Якщо домен є в цьому списку, він маркується під змінною threat_category. Залежно від того, до якої категорії належить домен у словнику, він маркується як malware, phishing або інші. Якщо ж домена немає в списку то

він маркується як safe. Варто зазначити, що такий спосіб не є абсолютно точним, оскільки не всі домени перевірені на безпечність. У цьому словнику зібрані лише найбільш поширені ресурси, що можуть розповсюджувати рекламу, збирати телеметрію та інше. Для забезпечення актуальності цієї додаткової функції застосовано автоматичне оновлення словника за допомогою PowerShell-скрипта, який кожні 300 секунд, оновлює вміст словника з Github-репозиторію. Також, цей скрипт переглядає вміст завантажуваного файлу, форматує його відповідно до вимог Logstash і вже потім зберігає його до словника [33, 34].

Наступною ланкою системи є вебінтерфейс для бази даних Kibana. Зв'язок з Elasticsearch встановлюється шляхом обміну інформацією між контейнерами всередині середовища Docker. Ідентифікація відбувається за заздалегідь прописаними іменами контейнерів в конфігурації Docker. Вебінтерфейс являє собою графічний користувацький інтерфейс із зручним управлінням базою даних через браузер замість стандартних запитів REST API. Тут можна так само налаштувати права доступу, як виглядатимуть потоки даних, керувати сховищем з логами та створювати візуалізацію.

Тут потрібно налаштувати лише подання інформації, а саме індекс для Data View. Індекси відображають те, що зберігається в базі даних. Оскільки раніше ми налаштували форматування як dns-logs-%{+YYYY.MM.dd}, то і індекс у Kibana буде виглядати dns-logs-*, де * означає усе що йде після. Таким чином ми налаштували відображення у вебінтерфейсі і тепер можемо проводити аналіз DNS-запитів і відповідей.

2.4 Висновки до розділу

В другому розділі було визначено необхідні складові системи та їх ролі у тестовій мережі. Таким чином було розроблено топологію мережі, що включає в себе:

– клієнтські пристрої, для генерації трафіку;

– Mikrotik CHR як основний резолвер мережі;

– роутер TP-Link, що виступає основним шлюзом між локальною та зовнішньою мережею;

– фільтр Logstash, що виконує функцію обробки Syslog-повідомлень від Mikrotik;

– Elasticsearch, що виконує функцію бази даних для мережі;

– Kibana вебінтерфейс для бази даних, для зручного керування, аналізу та перегляду проіндексованих логів;

– Docker, середовище контейнеризації де розгорнуто систему ELK Stack.

Виконано налаштування правил пересилання трафіку всередині віртуального маршрутизатора Mikrotik. Налаштовано контейнерне середовище Docker та розгорнуто програмний стек ELK Stack. Налаштовано роботу та правила індексації документів в базі даних Elasticsearch. Застосовано логіку парсингу Syslog-повідомлень від Mikrotik за допомогою використання плагіну grok в конфігураційному файлі фільтру Logstash. Прописано правила агрегації виокремлених змінних. Та відображення інформації з бази даних у вебінтерфейсі Kibana.

Після здійснених налаштувань прототип є повністю готовим до подальшого тестування.

3 ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ РОБОТИ СИСТЕМИ

3.1 Тестування роботи системи в звичайних умовах домашньої мережі

Першим етапом тестування буде аналіз роботи системи збору без сильного навантаження. Прототип буде імплементовано в середовище домашньої мережі. Оскільки, як DNS-резолвер ми використовуємо віртуальний Mikrotik CHR, тоді, для виходу в інтернет та для зв'язку з обладнанням провайдера нам потрібен фізичний шлюз. У тестовій мережі цим шлюзом виступає TP-Link AX-12, який отримує мережеві налаштування від провайдера автоматично, як DHCP-клієнт. Бюджетний роутер, для домашнього використання. Якби він мав функцію відправки логів на зовнішній ресурс, тоді б можна було задуматися над використанням цього роутера. Запускаємо окремо Mikrotik та систему збору і зберігання у Docker.

Після успішного запуску системи, ми бачимо, що логів немає у вебінтерфейсі та базі даних. Це відбувається тому, що на даний момент система працює лише якщо вручну вказати Mikrotik як DNS-сервер. Наприклад, за допомогою утиліти nslookup [35]. Вона дозволяє перевірити яку IP-адресу або будь-який тип DNS-записів, які має конкретний домен. Якщо запустити команду без вказування сервера, який використовуватиметься як DNS-сервер, тоді буде використовуватися сервер за замовчуванням, наприклад сервери інтернет провайдера. Результат роботи такої команди зображено на рис. 3.1.

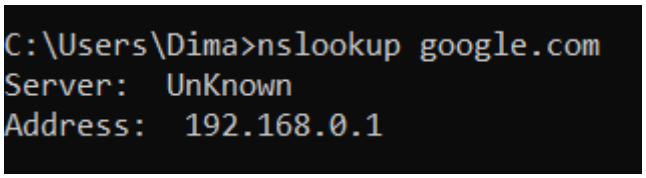


Рисунок 3.1 - Результат виконання команди nslookup без вказання DNS-сервера

Без вказування конкретного DNS-сервера, система використовує стандартний сервер, інформацію про який зазначено в налаштуваннях маршрутизатора, від якого пристрій отримав налаштування.

Також, повертаючись до налаштувань, які ми здійснювали в попередньому розділі. Система збору та зберігання налаштована так, що вона тільки слухає визначений порт 514. Коли Mikrotik перехоплює DNS-трафік з порту 53 TCP/UDP, він формує Syslog-повідомлення та надсилає його на 514 порт хостової системи, де розташований ELK Stack та сам Mikrotik CHR. Після того, як Mikrotik відправив дані, вони приймаються Docker. По внутрішній віртуальній мережі середовища, пакети передаються до контейнера Logstash. Він їх обробляє та розділяє текстові значення з логів на функціональні рядки, які потім використовуватимуться у вебінтерфейсі. Після того, як фільтр обробив дані, він надсилає їх у контейнер, де розташована база даних Elasticsearch.

Завдяки коректному налаштуванню параметрів мережевих інтерфейсів у VM VirtualBox для Mikrotik CHR, TP-Link, бачить Mikrotik як окремий пристрій в локальній мережі, не враховуючи, що він запущений у віртуальному середовищі ноутбука. Завдяки цьому, Mikrotik отримує окрему IP-адресу та мережеві налаштування, завдяки чому ми можемо вказати його як DNS-сервер.

Варто уточнити, що якби Mikrotik був фізичним пристроєм і виступав як основний шлюз до провайдера та зовнішньої мережі, то всі мережеві налаштування він би отримував конкретно від провайдера.

Після того, як Mikrotik, отримав IP-адресу, у даному випадку це 192.168.0.83. Ми можемо вказати його як DNS-сервер. Тоді команда nslookup буде виглядати наступним чином nslookup google.com 192.168.0.83, де:

– nslookup – це виклик основної команди;

– google.com – домен, який ми перевіряємо;

– 192.168.0.83 – DNS-сервер, який ми вказуємо, для того, щоб запит здійснювався тільки через конкретний сервер.

Результат роботи команди зображено на рис. 3.2. Як ми можемо бачити, в полі Address, вказано адресу віртуального роутера Mikrotik, а саме 192.168.0.83.

І тільки зараз, якщо ми поглянемо у базу даних та вебінтерфейс системи збору, ми побачимо, що там успішно записався лог DNS-запиту та відповіді. Також, оскільки ми налаштували правила агрегації даних, ми бачимо, що у

запиті вказано адресу ноутбука 192.168.0.207 та у відповіді також підв'язано адресу 192.168.0.207 за унікальним query_id події.

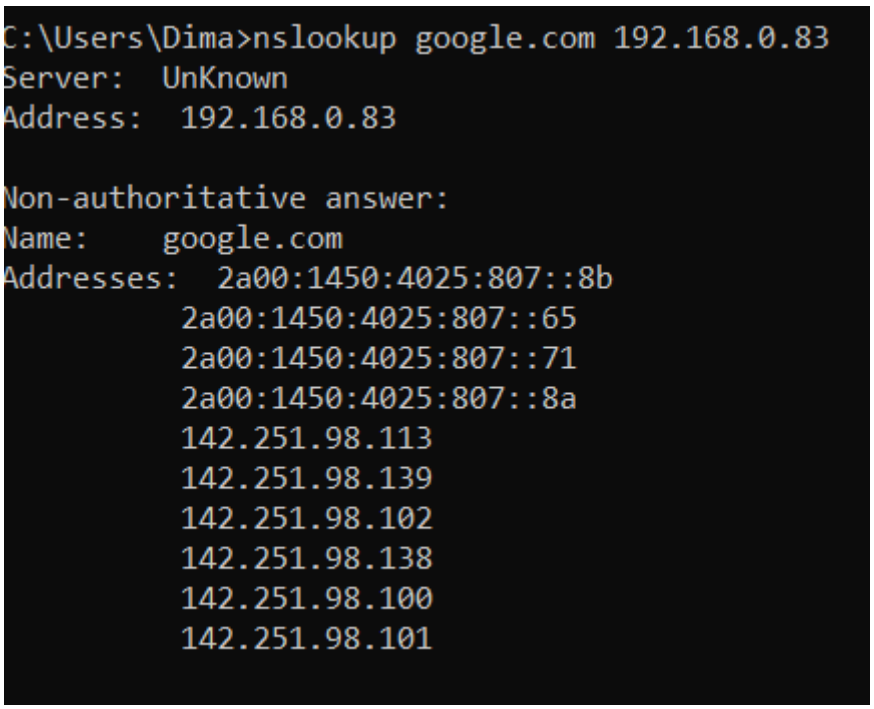


Рисунок 3.2 - Результат роботи команди nslookup з використанням адреси DNS-сервера

Це тестування показує, що система збору успішно перехоплює запит та пересилає його на фільтр Logstash, де він успішно обробляється, після чого ми бачимо запис у базі даних та, відповідно у вебінтерфейсі.

Також, потрібно перевірити чи система коректно обробить запит, що здійснюється до заздалегідь визначеного шкідливого домену з локального списку bad_domains.yml. Для цього, так само виконаємо запит із зазначенням Mikrotik, як DNS-сервера, але в параметрі домену, вкажемо ресурс з словника. Нехай цим ресурсом буде serasearchtop.com. В такому випадку, команда запиту виглядатиме наступним чином nslookup serasearchtop.com 192.168.0.83. На рис. 3.3 можна побачити результат успішного виконання команди.

Тепер, якщо ми перейдемо до вебінтерфейсу Kibana, ми побачимо, що запис про цю подію був успішно створений та був позначений як threat_category: malware. З чого можемо зробити висновок, що система повністю працює і реагує

на запити до заздалегідь визначених шкідливих доменних імен.

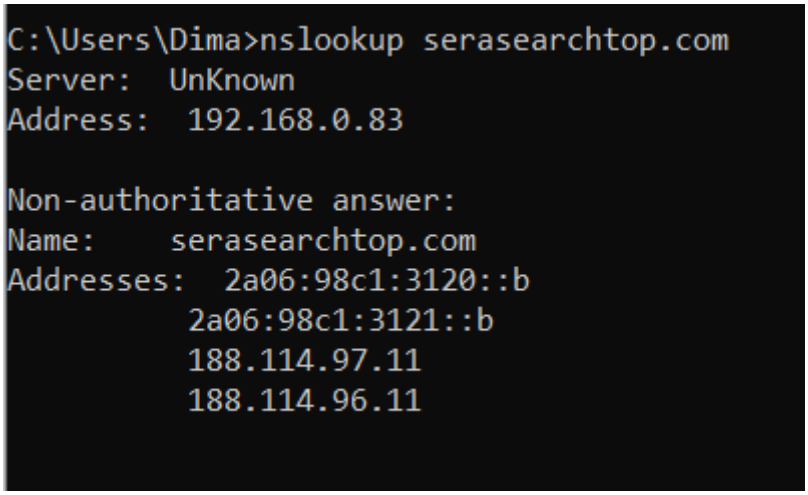


Рисунок 3.3 - Результат виконання команди nslookup із вказанням домену зі словника

Введення команди, це звичайно показник успішної роботи, але як система буде поводитися в реальній мережі з справжніми запитами від декількох пристроїв. Для того, щоб реалізувати такий сценарій, нам потрібно, щоб усі пристрої мережі використовували як DNS-сервер наш віртуальний Mikrotik. Є декілька способів, як це зробити. Перший з них це на кожному пристрої прописати як основний DNS- сервер IP-адресу маршрутизатора. Цей спосіб підійде, якщо мережа невелика, і пристрої в ній змінюються не часто. Але, у даному способі є великий недолік. Якщо якийсь пристрій просто змінить налаштування, до якого DNS-сервера йому звертатися, то в такому випадку він просто обійде Mikrotik і дані не будуть записані в базу даних.

Варто зазначити, що, якщо встановлений фізичний пристрій Mikrotik як основний шлюз між мережею та провайдером, тоді такий метод обходу системи не буде працювати. Тому що на Mikrotik ми налаштували раніше правила Redirect для NAT, які будуть завертати будь-який DNS-трафік, та пропускати його конкретно через Mikrotik. Це зроблено для того випадку, якщо пристрої вкажуть іншу адресу DNS-сервера в своїх налаштуваннях, то запити все одно будуть записані у базу даних.

У нашому ж випадку, ми маємо віртуальний Mikrotik. Який не виступає основним шлюзом до зовнішньої мережі, тому запити можуть проходити крізь нього. Для того, щоб уникнути ситуації, коли логи від якихось пристроїв не запишуться існує другий варіант налаштування. Це вказати IP-адресу Mikrotik, як основний DNS-сервер в налаштуваннях шлюзу до зовнішньої мережі.

Оскільки, у тестовій мережі таким шлюзом виступає роутер TP-Link, зайдемо в його налаштування та вкажемо там наш віртуальний маршрутизатор, як основний DNS-сервер. Оскільки, роутер TP-Link отримує налаштування динамічно від інтернет провайдера, то в налаштуваннях роутера вказані конкретні зовнішні DNS-сервери провайдера, які змінити ми в даному випадку не можемо. Якби підключення здійснювалося за допомогою задання статичних мережевих налаштувань, ми б могли вказати свої сервери там.

Для забезпечення належного функціонування нашої системи в межах локальної мережі роутера, нам достатньо лише налаштувати, щоб пристрої, всередині цієї мережі, зверталися до нашого Mikrotik. Враховуючи те, що роутер TP-Link роздає IP-адреси всім пристроям в локальній мережі, тобто працює як локальний DHCP-сервер, нам достатньо вказати IP-адресу Mikrotik, конкретно в налаштуваннях DHCP-сервера всередині роутера TP-Link. Вказуємо у полі Primary DNS, адресу Mikrotik, а саме 192.168.0.83. В такому випадку, після того, як кінцеві пристрої переотримають мережеві налаштування від роутера, вони побачать, що їм тепер потрібно вказувати адресу 192.168.0.83 як основний DNS-сервер. Завдяки чому всі запити будуть проходити конкретно через Mikrotik і, як наслідок, будуть логуватися, оброблятися та записуватися у базу даних.

В такому випадку, налаштування системи повністю готові для тестування. Отже, на даний момент рух трафіку по мережі виглядає наступним чином:

Клієнтські пристрої в підмережі 192.168.0.0/24, бажають отримати доступ до конкретних ресурсів. Спочатку перевіряють свій локальний кеш на наявність відповідних записів. Після цього вони переглядають налаштування які отримали від DHCP-сервера, що розташований на роутері TP-Link. В отриманих налаштуваннях вони бачать, що адреса основного DNS-сервера вказана

192.168.0.83. Це саме IP-адреса віртуального роутера Mikrotik. Клієнтські пристрої надсилають DNS-запити до цієї адреси.

Ці запити спочатку проходять до роутера TP-Link, який перенаправляє їх до Mikrotik CHR з адресою 192.168.0.83. Mikrotik отримує ці запити і завдяки правилу redirect пропускає їх спочатку через свою систему. Додатково, через правило логування на віддалену адресу, він відправляє логи про подію на ноутбук 192.168.0.207. Після чого надсилає DNS-запит назад до роутера TP-Link, оскільки він є основним шлюзом до зовнішньої мережі. Роутер TP-Link, в свою чергу перенаправляє ці пакети до заздалегідь отриманих DNS-серверів інтернет провайдера.

Переслані логи на адресу 192.168.0.207, на порт 514 фіксуються фільтром Logstash, який налаштований слухати всі події на порту 514 TCP/UDP. Після чого він їх обробляє, розділяє на складові частини.

Здійснення запиту, тобто запит з клієнтського пристрою проходить через Mikrotik і відправляється на віддалений DNS-сервер. Парсинг цього типу запиту в Logstash виглядає наступним чином:

– query_id ідентифікатор запиту;
– client_ip IP-адреса пристрою, що здійснив запит;
– client_port порт з якого здійснювався запит;
– domain ім’я запитуваного домену;
– query_type тип DNS-запиту;
– remote_server_ip адреса зовнішнього DNS-сервера;
– remote_server_port порт зовнішнього DNS-сервера;
– packet_action тип події, чи пакет надіслано (sent) чи прийнято (received).

Наступним чином відбувається обробка відповіді від зовнішніх DNS-серверів, від яких було отримано відповідь на запит з інформацією про запитуваний домен. Парсинг виглядає наступним чином:

– query_id ідентифікатор запиту. Потрібен для здійснення подальшої агрегації;

– resolved_ip отримана IP-адреса запитуваного домену;

– dns_error_message змінна для запису повідомлення помилки. Наприклад якщо домену не існує або вийшов час відповіді (timeout);

– domain дублюється назва домену;

– packet_action фіксація дії, що записана у пакеті.

Додатково, на цьому етапі створюються власні змінні для подальшої аналітики і агрегації.

Змінна threat_category, визначає тип безпечності домену. Порівняння базується на інформації з локального списку шкідливих доменів. Логіка полягає в тому, що з пакету запиту виокремлюється ім’я домену і порівнюється з наявним списком. Якщо дані збігаються, тоді Logstash маркує цей запит і подальшу відповідь на нього, наприклад як threat_category:malware. Якщо ж домену немає в цьому списку або він позначений як безпечний, тоді присвоює йому тег safe.

Далі блок aggregate. Він виконує функцію агрегації. Syslog за замовчуванням містить адресу пристрою, що зробив запит лише у запиті. У відповіді він не вписує запит. Щоб мати змогу співставляти адресу з доменом, було використано властивість Syslog. В обох випадках чи то запит чи відповідь, цей тип логу містить значення ідентифікатору, яке ми витягуємо за допомогою query_id. При отриманні логу запиту, Logstash виокремлює з нього query_id та client_ip. Після чого тимчасово зберігає client_ip і запускає таймер на 10 секунд. Якщо протягом цього проміжку часу надходить відповідь від сервера з таким же query_id, тоді він прописує в цей лог збережену адресу пристрою. І коли 10 секунд після початкового запиту спливають, він видаляє збережену адресу з пам’яті. Таким чином відбувається збагачення отриманої інформації.

Лише після того, як логи пройшли обробку у Logstash та мають структурований JSON-формат, вони надсилаються до бази даних Elasticsearch. Тут вони додатково індексуються та зберігаються.

Наступним кроком це вебінтерфейс Kibana, де ми можемо побачити зібрані логи в графічному інтерфейсі та проводити аналітику інформації з бази даних. Додатково є функція створення інформативної візуалізації, де можна

налаштувати наприклад, відображення IP-адрес, які надсилають найбільше запитів або ж відфільтрувати які адреси зверталися до конкретного домену. Таким чином вебінтерфейс стає фінальною точкою в шляху DNS-пакета від клієнтського пристрою або зовнішнього сервера до бази даних Elasticsearch.

Тепер можемо переходити до тестування прототипу системи у звичному режимі для домашньої мережі. Для початку, переглянемо список клієнтських пристроїв, які підключені до локальної мережі роутера TP-Link. Для цього будемо використовувати метод підключення до налаштувань роутера через вебінтерфейс. Роутер має адресу 192.168.0.1, тому для цієї дії в звичайному браузері, в даному випадку Google Chrome, виконуємо підключення до адреси роутера. Далі, переходимо у вкладку Network Map, де відображаються спрощені налаштування мережі. Після цього, у секції Clients ми можемо побачити адреси пристроїв в мережі, по якому саме стандарту бездротової мережі вони підключені та ім’я пристрою, якщо роутеру вдалося його визначити.

Як можемо бачити, на даний момент до мережі підключено 6 пристроїв. До того ж, тут не відображається основний ноутбук, оскільки з нього було здійснено підключення. Проте, його ми можемо побачити в налаштуваннях DHCP-сервера, зазначено, що йому видана адреса, що представлено на рис. 3.4.

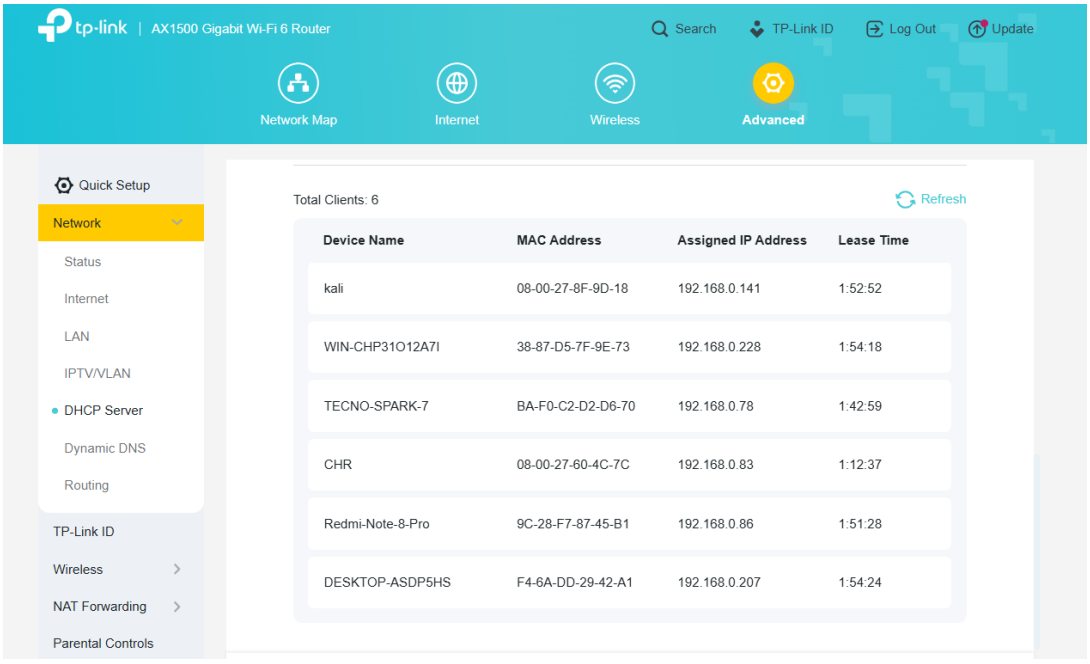


Рисунок 3.4 - Відображення підключених пристроїв в налаштуваннях DHCP-сервера

Загалом, пристрої в мережі мають такі IP-адреси:

– 192.168.0.83 Mikrotik CHR;

– 192.168.0.207 хостовий ноутбук для системи збору і Mikrotik;

– та інші клієнтські пристрої для генерації трафіку.

Переглянемо чи зберігаються індекси і записи у базі даних Elasticsearch. Для цього, здійснюємо підключення через браузер за адресою localhost із вказанням порту, на якому запущена база даних, в цьому випадку порт 9200. Також впишемо додаткові параметри для зручнішого відображення інформації. Вкажемо такі параметри _cat/indices?v&s=index:desc. Вони дозволяють переглянути записи в базі даних і завдяки s=index:desc будуть відсортовані за датою. Інтерфейс бази даних Elasticsearch із відповідними індексами зображено на рис. 3.5.

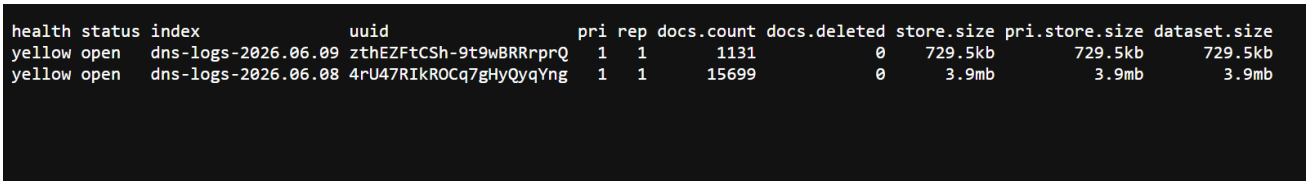


Рисунок 3.5 - Вебінтерфейс бази даних Elasticsearch

Як можемо бачити з рис. 3.5, інформація про конкретний індекс розбита на декілька стовпців, де:

– health визначає стан індексу. Він може бути green, yellow та red. В нашому випадку індекс має позначку yellow через особливості розгортання системи в середовищі Docker;

– status означає статус індекса. Іншими словами чи можна в нього записувати дані, читати чи змінювати;

– index назва індексу;

– uuid внутрішній ідентифікатор індекса в базі даних;

– pri відображає кількість шардів, тобто на скільки частин розбито документ, у нашому випадку 1;

– rep відображає кількість реплік, тобто скільки є ідентичних збережених

копій індекса, у нашому випадку є лише 1 копія;

– docs.count визначає кількість проіндексованих документів;

– docs.deleted визначає кількість видалених документів;

– store.size відображає об’єм дискового простору, який займає цей індекс;

– pri.store.size розмір оригінального індексу, без врахування шардів та реплік;

– dataset.size відображає об’єм дискового простору, який займає цей індекс включно з метаданими та іншими факторами.

З кожним новим логом, який потрапляє у базу даних Elasticsearch число docs.count та об’єм dataset.size будуть рости. Це меню є інформативним, для підтвердження, що індекси в базі даних створюються та у них записуються дані.

Для того, щоб переглянути які саме дані записуються в базу даних використаємо вебінтерфейс Kibana. В браузері переходимо за адресою локального хоста localhost з вказанням порту, на якому розгорнуто Kibana. В нашому випадку це порт 5601. Після відкриття інтерфейсу, потрібно впевнитися, що відповідні індекси було створено. Переходимо у вкладку Stack Management і у секції Kibana вибираємо пункт Data views. Ця вкладка визначає, як саме будуть подаватися дані з бази даних Elasticsearch у вебінтерфейсі Kibana. В цьому меню був попередньо створений індекс з назвою mikrotik з такими параметрами:

– Name mikrotik назва методу подання інформації;

– Index pattern dns-logs-* вигляд того, як буде подаватися інформація;

– Timestamp field @timestamp як буде відображатися часовий проміжок.

Оскільки вигляд подання інформації налаштовано, тепер ми можемо переглянути конкретно вміст логів, що зберігаються у базі даних. Для цього переходимо у вкладку Discover. В ній, ми можемо переглянути вміст конкретного логу, як у звичайному поданні, так і у форматі JSON. Можна здійснювати фільтрування, пошук та вибирати часовий проміжок, за який будуть відображені логи. Також, з лівого боку є секція з успішно розпарсеними полями. Це підтверджує, що фільтр Logstash успішно обробив та розпарсив повідомлення Syslog від Mikrotik на окремі функціональні поля. Наприклад,

зараз ми бачимо загальну кількість логів і можемо переглянути які саме адреси робили запити за допомогою поля client_ip. Як бачимо, що там вказані конкретні пристрої, які були відображені раніше в налаштуваннях роутера. А саме, такі пристрої як 192.168.0.207 та 192.168.0.86.

Факт того, що адреси пристроїв окремо відображаються та за ними можна здійснювати фільтрацію, означає, що конфігурація парсингу полів у Logstash є повністю коректною. Також можна відфільтрувати за доменами, де буде вказано повна текстова адреса домену та адреса пристрою який до нього звертався. Це також підтверджує, що правила агрегації працюють коректно та підв'язують адресу пристрою до запитуваного домену. Для додаткового підтвердження цього виконаємо команду nslookup без строгого вказання DNS-сервера. Тобто запит буде проходити по стандартному маршруту трафіку, без застосування додаткових параметрів та проходження нестандартних маршрутів. Домен для тесту буде google.com. І команда виглядає наступним чином nslookup google.com, де nslookup це виклик команди і google.com як додатковий параметр для виконання команди. Результат її виконання в консолі та в логах не повинен відображатися і в такому випадку це буде повноцінним підтвердженням коректної роботи прототипу системи моніторингу.

Результат тесту був виправданий та показав, що команда успішно виконується, тобто немає перешкод між пристроєм та сервером. Також, нам відображаються коректні результати виконання. Додатково бачимо фіксацію в логах роутера Mikrotik, а також відповідний запис у вебінтерфейсі Kibana, що продемонстровано на рис. 3.6. Тобто, для стандартних запитів система повністю адаптована.

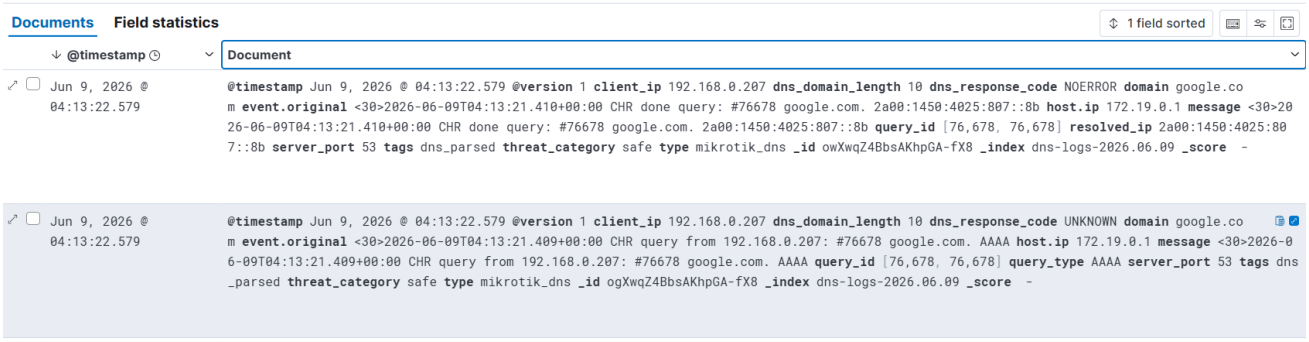


Рисунок 3.6 - Відображення логів у вебінтерфейсі Kibana

У вкладці Discovery бачимо потік логів від конкретних пристроїв, що підключені до основного роутера мережі. Це означає, що активність усіх пристроїв в мережі фіксується коректно та без перешкод. Варто зазначити, що підписки типу free на Mikrotik CHR з обмеженням пропускної здатності портів в 1 мегабіт на секунду повністю вистачає для збору, обробки та логування DNS-трафіку звичайної локальної мережі.

3.2 Симуляція атаки DNS Amplification для тестування роботи системи в нетипових умовах

Оскільки тестування системи в звичайному режимі показало позитивні результати, настав час перевірити роботу мережі під час нетипових умов. Також, це тестування спрямоване оцінити чи витримає система навантаження та чи вистачить підписки free для Mikrotik CHR з обмеженням пропускної здатності до 1 мегабіта на секунду.

Для симуляції атак, вирішено використовувати образ Kali Linux [36], що розгорнутий у віртуальному середовищі гіпервізора VirtualBox. Дистрибутив Kali Linux було обрано через те, що він невимогливий до ресурсів хостової машини та має весь необхідний інструментарій для реалізації тестування, без необхідності додаткового встановлення. Для початку, налаштуємо віртуальну машину з системою. Для цього використаємо офіційний образ системи з офіційного сайту дистрибутива. У VirtualBox проходимо стандартне налаштування системи, таке як виділення оперативної пам'яті та об'єм дискового простору. При інсталяції образу на віртуальний носій, потрібно додатково вибрати параметр, який відповідає за завантаження необхідних інструментів ще при інсталяції системи. Завдяки цьому не буде потреби в додатковому завантаженні інструментів.

Першим методом тестування системи буде симуляція атаки DNS Amplification [37, 38, 39]. Цей вид атаки використовує базові принципи роботи

DNS. Запит DNS є невеликим за обсягом, наприклад 50 байт. Тоді як відповідь на запит типу ANY, тобто всі наявні типи DNS-записів, може бути і декілька тисяч байтів. Саме на цьому принципі і базується атака з посилення DNS. Тобто, хід атаки виглядає наступним чином:

– атакуючий робить DNS-запити до серверів підміняючи свою IP-адресу на адресу пристрою жертви. Оскільки DNS також працює по протоколу UDP, а специфіка його полягає в тому, що непотрібно встановлювати пряме та надійне з'єднання, IP-адресу можна легко підмінити. Далі, у сервера запитуються записи типу ANY, тобто всі наявні записи, такі як A, AAAA, MX, NS, TXT та інші. Запит має обсяг пам'яті приблизно у 50 байт;

– наступним і головним кроком є отримання відповіді від сервера. Відповідь на запит типу ANY може сягати декілька тисяч байт. Враховуючи розмір запиту, відповідь просто перевантажує канал жертви, оскільки його пропускна здатність може бути низькою;

– коефіцієнт посилення вираховується так: розмір відповіді порівнюється з розміром запиту і частка від ділення цих складових дорівнює коефіцієнту посилення атаки. Вважається, що більший коефіцієнт дорівнює набагато більшим наслідкам для жертви від цієї атаки.

Основне налаштування, яке необхідно зробити для віртуальної машини з Kali Linux, це вказати в налаштуваннях VirtualBox тип мережевого адаптера міст. Це робиться для того, щоб віртуальна машина могла виходити в інтернет з окремою IP-адресою. Тому що якщо цього не зробити, вона буде надсилати запити з адреси хостової машини.

В даному випадку схема атаки виглядає так:

– ноутбук з адресою 192.168.0.207 виступає в ролі жертви;

– віртуальна машина з адресою 192.168.0.141 виступає в ролі атакуючого;

– роутер Mikrotik з адресою 192.168.0.83 виступає в ролі основного резолвера для мережі;

– ELK Stack, що розташований на хостовому ноутбуці з адресою 192.168.0.207, буде фіксувати усю мережеву активність всередині мережі.

Оскільки, всі пристрої налаштовані, окрім дистрибутива Kali, потрібно зробити деякі налаштування для змоги проведення атаки. Для цього був написаний скрипт на мові програмування Python. Вона була вибрана в міру своєї простоти. Для цього скрипта підключатиметься бібліотека scapy.all та sys за допомогою модуля import [40]. Для ідентифікації машини, на яку буде здійснюватися атака, адреси резолвера та доменного імені, було введено наступні змінні:

– VICTIM_IP змінна, в якій вказується адреса машини жертви;

– RESOLVER_IP змінна, в якій вказується адреса DNS-резолвера;

– TARGET_DOMAIN змінна, в якій вказується доменне ім’я ресурсу.

В нашому випадку VICTIM_IP це ноутбук 192.168.0.207, RESOLVER_IP це віртуальний маршрутизатор Mikrotik та TARGET_DOMAIN для тесту буде google.com. Після того, як основні параметри для здійснення атаки визначені, була прописана логіка підміни IP-адреси для заголовку UDP-пакету та підставлення туди адреси жертви, в даному випадку 192.168.0.207. Цей крок є одним із основних в побудуванні атаки. Оскільки це дозволяє перенаправити DNS-відповіді від зовнішніх серверів на цільову адресу. Також, в заголовок пакету підставляється адреса резолвера 192.168.0.83 та визначено правило рекурсії для пошуку доменного імені. Це зроблено для того, щоб більш прозоро показати хід цієї атаки в системі моніторингу. Після чого, за допомогою внутрішніх команд бібліотеки Scapy збирається UDP-пакет і відправляється в циклі, для імітації великої кількості трафіку.

Після генерації пакета, він відправляється мережевою картою на вказаний DNS-резолвер, який, в свою чергу, відправляє цей пакет на основний шлюз. Цільовий сервер бачить запит, обробляє його та надсилає велику за обсягом відповідь. Це можливо завдяки тому, що у налаштуванні скрипта вказано тип записів 255 тобто ANY. Після чого сервер відправляє назад об'ємну відповідь. І під час обробки цієї відповіді роутером TP-Link або резолвером Mikrotik, виникає ризик нестачі пропускної здатності, оскільки відбувається забиття каналу сміттєвим трафіком. Внаслідок чого, може виникнути відмова в

обслуговуванні, на що і спрямована цей тип атаки. Візуалізацію логів під час її проведення наведено на рис. 3.7.

Рисунок 3.7 - Сплеск логів з записами типу ALL після здійснення атаки

В результаті, в системі моніторингу ми бачимо різке зростання трафіку з однієї IP-адреси, конкретно з 192.168.0.207. Що свідчить про роботу вищезазначеного скрипта та проходження атаки. Можемо також бачити, що не всі пакети проходять у базу даних. Також, оскільки атака відбувається на основі протоколу UDP, втрата пакетів ніяк не фіксується і не здійснюється спроби повторної відправки пакетів сервером. Втрата пакетів може відбуватися з декількох причин:

– низька пропускна здатність роутера Mikrotik. Оскільки для прототипу системи використано безкоштовну підписку з обмеженням пропускної здатності в 1 мегабіт на секунду, це може бути причиною втрати деяких пакетів;

– захист від DoS або DDoS атак на основному роутері TP-Link. Деякі роутери можуть мати захист від таких типів атак. Захист відбувається наступним чином: якщо роутер фіксує велику кількість трафіку на одну IP-адресу пристрою, він просто блокує або відкидає частину пакетів, щоб не допустити перевантаження каналу;

– налаштування брандмауера операційної системи Windows. Також може бути причиною втрати пакетів, оскільки механізм роботи в нього схожий до захисту від даних типів атак роутера TP-Link.

Система зібрала близько 2000 логів про атаку, коли під час самої атаки

було відправлено приблизно 10000 запитів. Оскільки деякі запити могли просто не дійти до цільового сервера або ж втратитися по дорозі назад, вважаю такий показник абсолютно прийнятним для тестового прототипу. Оскільки ціль даної кваліфікаційної роботи побудувати систему збору логів.

3.3. Симуляція атаки DNS Subdomain Enumeration

Наступна атака є одним з найперших етапів проведення більш складних атак. Перед тим як сканувати на вразливості або ж здійснювати компрометацію конкретного ресурсу, спочатку потрібно просканувати його на наявність субдоменів. Це може бути наприклад dev.google.com, accounts.google.com. education.google.com і так далі. Під час даного типу атаки генерується велика кількість трафіку, оскільки надсилається багато запитів до можливих субдоменів та отримується багато відповідей якщо вони існують або ж ні.

Розберемо атаку на прикладі домену google.com. Хід атаки виглядатиме наступним чином:

– атакуючий, використовуючи наявну утиліту або ж спеціалізований скрипт, перебирає дані зі словника в якому записані можливі адреси;

– відправляє запит до кожного з можливих субдоменів зі списку;

– отримує відповідь про наявні субдомени або ж їхню відсутність. Завдяки цьому можна звузити вектор атаки або знайти вразливіше місце;

– після визначення відбувається сама атака або подальший збір інформації.

В нашому випадку ми пройдемо лише кроки з аналізу наявності субдоменів, без проведення атак, оскільки це виходить за межі теми кваліфікаційної роботи та аналіз субдоменів проводиться виключно в рамках тестування прототипу системи.

Для проведення симуляції атаки на сканування субдоменів буде використовуватися утиліта fierce, що вбудована в дистрибутив Kali Linux. Ця

утиліта використовує два методи розвідки субдоменів. Виконувана команда якої виглядає наступним чином fierce --dns-servers 192.168.0.83 --domain google.com, де:

– fierce виклик самої команди;

– параметр –dns-servers вказує конкретний DNS-сервер, через який будуть проходити запити. В нашому випадку вказана IP-адреса Mikrotik 192.168.0.83;

– параметр –domain вказує, який саме основний домен буде скануватися на наявність додаткових субдоменів. В нашому випадку це домен google.com.

Перший і основний спосіб це спроба передачі зони за допомогою запиту AXFR [41]. Існує він для того, щоб адміністраторам доменів було зручно отримати список всіх субдоменів компанії для подальшої обробки цих субдоменів або ж внесення в базу даних. При правильному налаштуванні, сервери відхиляють запит AXFR від зовнішніх адрес. Тобто цей тип запиту доступний лише за локальним списком доступу, які налаштовують адміністратори.

Саме помилку в конфігурації шукає утиліта fierce. Тобто, спочатку вона надсилає AXFR-запит до основного домену, після чого, якщо є помилка в конфігурації дозволів, сервер надсилає у відповідь список субдоменів та підключених пристроїв, що значно полегшує розвідку перед здійсненням атаки. Сучасні сервери здебільшого налаштовані відхиляти запити типу AXFR. Тому утиліта fierce застосовує другий метод розвідки субдоменів що належать до основного домену.

Перед переходом до наступного методу сканування субдоменів, утиліта fierce робить перевірку налаштувань DNS-сервера. Оскільки він може бути налаштований так, що на будь-який випадковий субдомен, буде надсилатися одна й та сама відповідь. Для перевірки ця утиліта використовує випадково згенерований субдомен, наприклад hoiqgqiwg.google.com, для перевірки на наявність цього методу захисту від подібних атак.

Другий метод – це перебір, або ж bruteforce, локального або зовнішнього словника з можливими іменами субдоменів. В цьому списку прописані тисячі

комбінацій адреси субдомену та основного домену. До прикладу dev.google.com, де dev це субдомен, google ім’я головного домену та .com ім’я доменної зони, до якої належать всі інші типи доменів. Під час перебору кожного запису зі списку, атакуюча машина здійснює DNS-запит типу A, тобто запит на отримання адреси типу IPv4. Після здійснення запиту, чекає на відповідь від сервера. В такому випадку, коли відбувається перебір імен субдоменів, відповідей від сервера може бути два типи, такі як:

– статус NOERROR, який означає, що домен існує і немає помилки під час формування відповіді сервера;

– статус NXDOMAIN, який означає, що домена не існує і сервер повертає відповідну помилку, а саме Non-Existent Domain.

Відповідно до відповідей сервера, утиліта маркує субдомен відповідно як існуючий і повертає його визначену IP-адресу. Якщо ж домену не існує, то відповідно пропускає його та лишає без адреси. Виведення результату роботи утиліти здійснюється через консольний рядок, де відображається обробка кожного варіанту можливого субдомену, що детально показано на рис. 3.8.

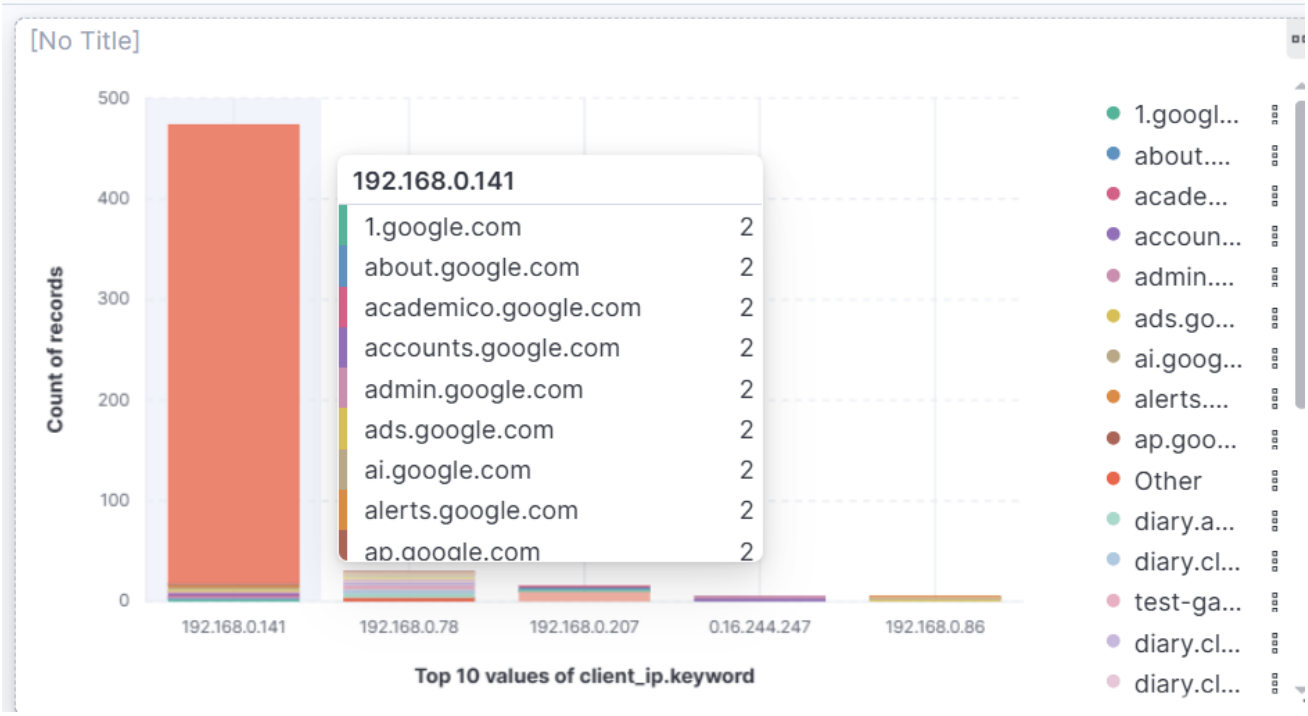


Рисунок 3.8 - Здійснення запитів з адреси до різних субдоменів

Під час роботи команди ми можемо бачити в вебінтерфейсі сплеск логів з адреси Kali Linux 192.168.0.141. Запити логуються коректно та відповідно до заздалегідь описаних правил. Бачимо, що навіть при такому потоку запитів не спрацьовують правила захисту від DDoS-атак або подібних їм видів атак. Через що у базу даних записуються всі запити та відповіді під час симуляції. Явним маркером конкретно атаки, спрямовану на розвідку потенційних варіантів субдоменів, є сплеск появи логів з помилкою DNS типу NXDOMAIN. При звичайному користуванні інтернетом, мається на увазі без здійснення кастомних запитів, а при звичайному перегляді вебсторінок і так далі, ймовірність появи типу помилки NXDOMAIN, якщо не фактично неможлива, то шанс її появи дуже близький до цього. Велика кількість записів із помилками NXDOMAIN зафіксована системою, що показано на рис. 3.9.

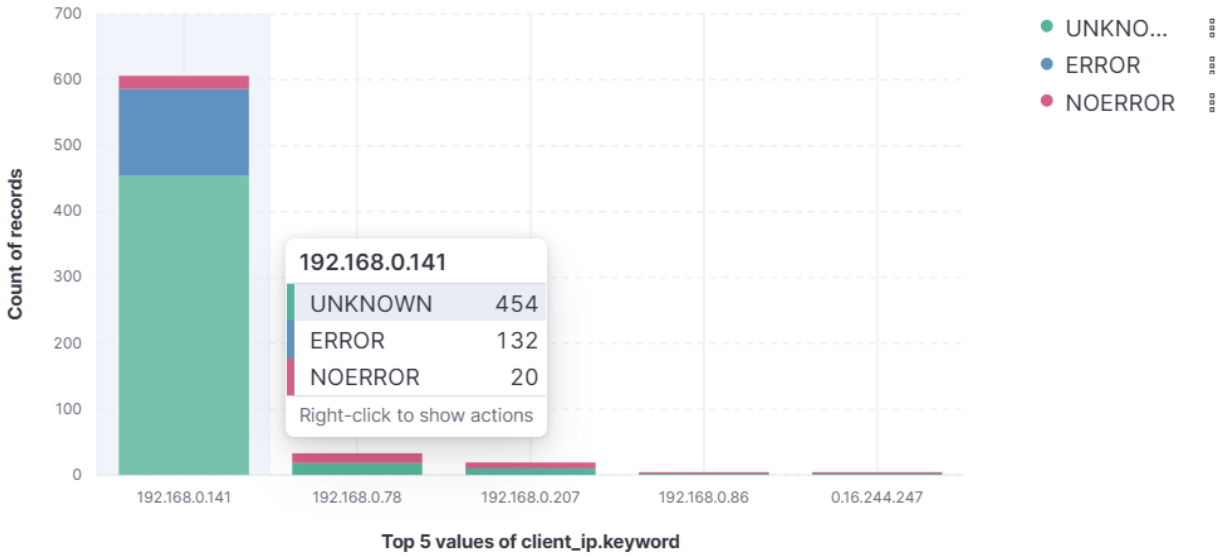


Рисунок 3.9 - Велика кількість DNS-відповідей з помилками

В логах ми якраз і бачимо цей індикатор: сплеск логів з помилкою NXDOMAIN. Це свідчить, що система успішно прийняла, обробила та відправила на зберігання логи під час високого навантаження на мережу. При цьому, варто зазначити, що обмеження пропускної здатності інтерфейсів Mikrotik до 1 мегабіта на секунду повністю вистачає для перехоплення такої інтенсивності нетипового трафіку.

3.4 Аналіз роботи системи з вбудованим правилом перевірки DNS-запитів до шкідливих доменів

В цьому пункті ми розглянемо прописану функцію у файлі конфігурації фільтру Logstash. Ця функція виконує виокремлення з тексту логу ключового імені домену та подальшим його співставленням з локальним списком шкідливих доменів. Цей список оновлюється кожні 300 секунд з відкритого репозиторія на Github. Автооновлення списку шкідливих доменів реалізовано за допомогою скрипта Powershell [42]. Логіка роботи цього скрипта:

– скрипт автоматично активується кожні 300 секунд;

– він ініціює з’єднання з репозиторієм за допомогою стандартних конструкцій Powershell;

– скрипт завантажує актуальний, на момент з'єднання, файл з репозиторія;

– виконує форматування вихідного файлу відповідно для уніфікації формату для фільтрів Logstash;

– фінальним етапом він записує зміни у файл bad_domains.yml. Саме цей файл вказаний у налаштуваннях фільтрів Logstash і звідки він порівнює отримані домени.

Варто додатково зазначити, що для реалізації цієї логіки, використовується репозиторій Github від StevenBlack. Він вважається авторитетним у спільноті для локального чи корпоративного використання. Оскільки він автоматично оновлює дані у файлі опираючись на дані з інших авторитетних джерел, які займаються аналізом та перевіркою доменів.

Для того, щоб протестувати чи коректно працює прописана логіка ми вручну виконаємо DNS-запит на домен зі списку. Для тестового варіанту було вибрано домен ssl.googleanalytics.com. Оскільки для тестування він є найбільш безпечним і до цього списку потрапив лише через надмірний збір телеметрії.

Отже, команда виглядатиме наступним чином nslookup ssl.googleanalytics.com. Де nslookup є викликом команди, а ssl.googleanalytics.com доменом для тестування. Результат виконання у командному рядку Windows

відобразив коректну відповідь від DNS-сервера. Це свідчить про те, що запитуваний домен є активним та інформація про нього зберігається в мережі.

У вебінтерфейсі Kibana з'явився відповідний запис в полі threat_category з параметром malware, що зображено на рис. 3.10. Це значення присвоєно цьому домену відповідно до мітки у локальному словнику шкідливих доменів.

Рисунок 3.10 - Відображення коректного співставлення небажаного домену

Запис повністю відповідає тому, що показав результат роботи команди nslookup. З цього, можна зробити висновок, що локальна перевірка домену на шкідливість або небажаність за заздалегідь вказаним списком повністю працює.

3.5 Тестування системи в умовах мобільної мережі

Метою цього пункту є тестування та аналіз роботи системи в відмінному середовищі від умов домашньої мережі. Для реалізації даного етапу тестування, буде використовуватися мобільна мережа, що буде роздана з мобільного пристрою на ноутбук, на якому хоститься система.

Мобільний пристрій, з якого відбувається поширення мобільної мережі не

підтримує додаткові налаштування параметрів DNS-сервера. Тому, для того, щоб Mikrotik бачив трафік, потрібно, на кожному підключеному пристрої вручну вказати в налаштуваннях адресу резолвера. В даному тестуванні схема підключення виглядає наступним чином:

– мобільний пристрій, який підключений до базової станції мобільного оператора, створює свою мережу до якої будуть підключатися пристрої;

– ноутбук, що буде здійснювати підключення до мережі мобільного пристрою.

Оскільки, визначено схему підключення та конкретні пристрої які туди входять, через зміну умов мережі, потрібно внести зміни в налаштування програмного забезпечення. Оскільки відбулась зміна схеми отримання локальної адреси, потрібно заново налаштувати систему збору та моніторингу відповідно до нових умов. Головні налаштування, які потрібно зробити, це дізнатися нові адреси пристроїв у мережі, а конкретно хостового ноутбука та віртуального роутера Mikrotik. Для початку, в налаштуваннях роутера було оновлено налаштування дії пересилання логів. Замінено значення в полі remote address на поточну адресу ноутбука. Тепер Mikrotik буде надсилати логи назад до хостового ноутбука. Це забезпечить їх безперебійне проходження через систему обробки, індексації та зберігання отриманих логів.

Далі, необхідно налаштувати передавання мережевої активності з ноутбука до резолвера Mikrotik. Для цього було прописано статичні налаштування на ноутбуці для протоколу IPv4 в налаштуваннях мережевого адаптера пристрою. Після застосування зазначених налаштувань виконуємо тестовий запит команди nslookup, без вказування конкретного резолвера.

Результатом виконання є успішна відповідь від сервера та записані логи у базу даних. Це підтверджує як можливість гнучкого налаштування мережі під різні види мережі, так і високу мобільність для використання системи в умовах різних типів умов експлуатації.

3.6 Рекомендації стосовно подальшого покращення роботи системи

Метою даного підрозділу є аналіз виявлених недоліків системи та надання рекомендацій для їх покращення.

Для початку, варто звернути увагу на Mikrotik CHR. Для реалізації прототипу було обрано саме віртуальне рішення, а не фізичний пристрій. Проте, для реалізації покращення системи збору, варто звернути увагу на апаратні рішення від Mikrotik. Оскільки вони не потребують окремої підписки для належного функціонування, та обмежені лише своїми технічними характеристиками. Але, варто зазначити, що використання конкретно апаратного рішення для DNS-резолвера мережі зменшить мобільність системи.

Наступним кроком для покращення системи запропоновано перегляд наявної конфігурації Logstash. А саме grok-патернів та правил агрегації. Оскільки, для тестового прототипу достатньо базових налаштувань системи та простої логіки агрегації. Для реальної інтеграції в мережу можуть знадобитися додаткові параметри grok, залежно від потреб та вимог до безпеки конкретної мережі в якій розгортатиметься система моніторингу. Або ж використання додаткових плагінів Logstash, наприклад як GeoIP. Функція якого дозволяє додатково налаштувати grok-патерни на парсинг та ідентифікацію інформації про географічне розташування запитуваних серверів. Підключення цієї функції дозволить аналізувати та виявляти конкретно в які географічні зони надсилаються запити. Це допоможе ідентифікувати запити до потенційних C2-серверів або ж контрольних доменів ботнетів. Детальніша перевірка яких, може забезпечити збільшення безпеки мережі.

На даний момент вебінтерфейс Kibana налаштовано лише для відображення індексованих логів, фільтрування їх за ключовими полями та створення візуалізації за ними. Для функціонування тестового прототипу не було потреби налаштовувати права доступу та ролі для користувачів. Але при подальшому розгортанні або покращенні системи, функція прав доступу і моніторингу дії користувачів з базою даних може знадобитися для того, щоб

проводити належний аудит безпеки всередині команди аналітиків інформаційної безпеки. Для збільшення ступеня захищеності системи, в подальшому, потрібно використовувати шифрування трафіку. В Kibana є доступні рішення з цього приводу. Наприклад як генерація SSL-сертифікату та вибір типу шифрування даних, щоб вони не передавалися у відкритому вигляді. Це також дозволить захистити конфіденційну інформацію від атак типу Man-in-the-middle. Оскільки дані будуть надійно зашифровані, здійснення такого типу атак може бути недоцільним для зловмисників, через неможливість підбору вихідних даних для розшифрування інформації.

Також, корисним доповненням до системи може виступити блок, для визначення та відображення географічного розташування запитуваних серверів, про який було зазначено вище.

Наступною рекомендацією стосовно покращення системи є використання агентів зі збирання інформації. ELK Stack має в собі функцію для встановлення незалежних, невеликих агентів під назвою PacketBeat. Вони встановлюються на кінцеве обладнання, моніторинг мережевої активності якого потрібно здійснювати. З огляду на її невелику вагу для файлової системи, встановлення агента не спричинить проблем з нестачею пам’яті. Функціонал агента полягає в тому, що він збирає інформацію про мережеву активність пристрою та відправляє цю інформацію на віддалений сервер, де розташована система моніторингу ELK Stack. Після чого проводиться стандартна обробка за налаштованими правилами у Logstash. Функція незалежних агентів PacketBeat дає додаткову можливість збирати інформацію з пристроїв, які не розташовані в одній мережі із сервером обробки. Що дозволяє значно розширити зону спостереження за інформаційною безпекою наприклад співробітників.

Також, варто зазначити, що цю систему можна використовувати не тільки для зберігання логів мережевої активності та DNS-запитів. Logstash підтримує різні методи налаштування фільтрів та виокремлення подій. Також, окрім агентів збору мережевого трафіку PacketBeat, є ще агенти з аналізу файлової системи FileBeat та багато інших типів агентів, розроблених для здійснення

збору даних тої чи іншої області аналізу.

База даних Elasticsearch може зберігати та індексувати інформацію про будь-які події, незалежно чи це будуть дампи активності пристроїв в мережі, чи зберігання логів про фінансові трансакції. Ця властивість дає можливість створювати окремі індекси для кожного з типу відстежуваної інформації та зберігати або обробляти централізовано.

одатково, варто зазначити, що розроблений прототип системи використовує лише один кластер бази даних Elasticsearch. Для мереж великого масштабу варто використовувати декілька кластерів бази даних та мати резервну копію всієї зібраної інформації.

Для того, щоб не засмічувати дисковий простір старими логами, варто налаштувати періодичне видалення давніх логів про мережеву активність. Наприклад, відповідно до стандарту ISO 27001, термін зберігання мережевих слідів конкретно не визначено [20, 21]. Кожна компанія визначає цей термін відповідно до законодавства, сфери діяльності та оцінки ризиків. Практичний оптимальний термін зберігання логів становить 12 місяців з моменту їх створення. Де перші 90 днів, логи зберігаються у базі даних для швидкого перегляду інформації в них, а на подальший період вони архівуються та стискаються для зменшення використовуваного об’єму дискового простору.

Дотримання вищезазначених рекомендацій, дозволить забезпечити значне покращення системи та дозволить ефективно реагувати на інциденти інформаційної безпеки в мережі.

3.7 Висновки до розділу

Результати тестування показали, що система повністю коректно працює, а саме збирає, обробляє, індексує та зберігає логи DNS-трафіку у базі даних. При невеликому навантаженні в умовах домашньої мережі система працює без збоїв та системних помилок. Коректно визначає та виділяє IP-адреси з повідомлень

Syslog від Mikrotik. Також, під час тестування визначено, що обмеження пропускної здатності для безкоштовного типу підписки Mikrotik CHR повністю вистачає для роботи з високою відмовостійкістю.

Було протестовано роботу прототипу в умовах нетипової мережевої активності в мережі. Нестандартними умовами стало здійснення симуляції декількох видів атак, включно з перевіркою відображення при імітації компрометації пристрою в локальній мережі.

Атака з примноженням DNS Amplification. Ця симуляція цього типу атаки була спрямована на перевірку відмовостійкості прототипу системи в умовах великої кількості об’ємних DNS-відповідей від зовнішніх серверів. Симуляція показала, що система повністю коректно збирає, обробляє та зберігає DNS-трафік. Проте, виявлено, що не всі пакети протоколу UDP надходять до системи. Це відбувається внаслідок налаштувань роутера TP-Link та політиками безпеки операційної системи Windows. Також, спричиняти втрату пакетів можуть заходи безпеки зі сторони інтернет провайдера, зовнішнього домену, до якого здійснювалися запити або ж звичайною нестачею пропускної здатності інтерфейсів віртуального маршрутизатора Mikrotik в межах безкоштовної підписки.

Наступною здійсненою симуляцією атаки була атака на здійснення розвідки цільового домену на наявність активних субдоменів. Тестування показало, що система успішно фіксує та зберігає логи про мережеву активність пристрою, який надсилає DNS-запити до цільового субдомену. Навіть при великому навантаженні, правила фільтрування та агрегації, що були закладені в конфігураційному файлі Logstash успішно спрацювали та обробили кожен лог, що був надісланий до фільтру. Вебінтерфейс Kibana показав на візуалізації різкий сплеск запитів до різних субдоменів і стрімке зростання отримання помилки NXDOMAIN, що свідчить про успішну ідентифікацію симульованої атаки.

Додатково, було протестовано функціонал співставлення запитуваних ресурсів зі списком шкідливих доменів з авторитетного репозиторія на Github.

При штучному надсиланні запиту до одного з доменів зі списку, система успішно обробила цей запит та коректно додала відповідне маркування threat_category до логів, що пов’язані з цим конкретним доменом. Тобто, використання цієї логіки є повністю доцільним для додаткового забезпечення захисту локальної мережі.

З вищеперелічених результатів тестування, було зроблено висновок, що прототип є повністю функціональним та готовим до використання для моніторингу мережевої активності пристроїв у мережі. Прототип виконує частину функцій реальної SIEM-системи, що використовується в справжніх умовах корпоративної мережі різних компаній. Також, створений прототип відкриває можливості для подальшого його удосконалення у повноцінну систему моніторингу мережевих подій.

Також, надано рекомендації, стосовно подальшого покращення системи моніторингу, для її перекласифікації у повноцінну систему виявлення та реагування на інциденти SIEM. Зазначено виявлені поточні недоліки системи та можливі подальші обмеження. Розроблено рекомендації стосовно підвищення рівня захищеності мережі.

ВИСНОВКИ

В рамках виконання завдання кваліфікаційної роботи по побудові системи централізованого журналювання DNS-запитів побудовано повністю працюючий та готовий до інтеграції або розширення у мережі, де необхідно збільшити спостереження за мережевим трафіком та зменшити ризики інформаційної безпеки.

На початку роботи було обґрунтовано актуальність теми із зберігання та аналізу логів мережевої активності пристроїв мережі, як один із інструментів забезпечення інформаційної безпеки мережі. Також зазначено актуальність захисту мережі від шляхів отримання шкідливого програмного забезпечення чи можливості ідентифікації загроз та здійснення превентивних дій, для недопущення втрати конфіденційної інформації, через неавторизовані запити до C2-серверів зловмисників.

В рамках першого розділу було розглянуто логіку та специфіку роботи протоколу DNS, його переваги та недоліки. Було визначено основні вектори атак на протокол доменних імен та вимоги для протидії цим загрозам. Великою частиною цього розділу було присвячено аналізу наявних та актуальних рішень, які використовуються компаніями для забезпечення безпеки та детального аналізу мережевого трафіку. Вибрано програмне забезпечення для реалізації прототипу та визначено подальші кроки для подальшої реалізації.

В другому розділі розроблено повністю прототип системи та виконано налаштування як мережевого обладнання на базі віртуального маршрутизатора Mikrotik CHR. Також налаштовано логіку маршрутизації трафіку у локальній мережі. Основну частину було присвячено налаштуванню системи обробки та зберігання DNS-запитів на базі програмного стеку для парсингу та зберігання даних ELK Stack. Додатково було налаштовано правила обробки та збагачення логів за допомогою конфігурації агрегації фільтру Logstash. В цьому розділі створено повноцінний прототип для подальшого тестування в третьому розділі.

Як вже було зазначено, третій розділ повністю присвячений тестуванню

коректної роботи та відмовостійкості розробленого прототипу системи централізованого журналювання DNS-запитів. Було проведено тестування в умовах функціонування звичайної мережі. Під час цього тестування система коректно обробила адреси пристроїв, проіндексувала захоплені повідомлення та успішно виконала правила агрегації та співставлення адрес пристроїв із запитами та відповідями на них.

У наступних етапах тестування було проведено аналіз роботи системи під час нетипових умов функціонування мережі. Було здійснено симуляції атак на DNS-протокол. При здійсненні атаки на перевірку відмовостійкості системи було виявлено, що система успішно обробила значну кількість великих за об’ємом відповідей від зовнішніх DNS-серверів.

Також, було здійснено симуляцію розвідки зовнішнього домену та аналіз реакції системи на таку мережеву активність. Цей тип тестування дав розуміння, що система без помилок фіксує значну кількість як вихідних запитів від внутрішніх пристроїв так і вхідну кількість відповідей з помилками від зовнішніх доменів.

Додатково, було проаналізовано функціонал зі співставлення та маркування запитів до шкідливих доменів за заздалегідь визначеним списком небажаних доменів з авторитетного репозиторія.

Під час виконання роботи було проаналізовано та підтверджено актуальність теми захисту мережі. Розроблений прототип показав, наскільки важливо вчасно ідентифікувати та реагувати на інциденти інформаційної безпеки навіть у звичайній домашній мережі.

Також, результатом роботи став повністю функціональний прототип, який коректно фіксує логи мережевої активності, для подальшого її аналізу. Поставлене завдання було виконано. Це підтверджує етап тестування, де система показала свою стійкість та коректність роботи не лише у звичайних умовах мережі, а й при виникненні атак на мережеву інфраструктуру.

ПЕРЕЛІК ДЖЕРЕЛ

1. ENISA. ENISA Threat Landscape 2024: Distributed Denial of Service (DDoS) [Електронний ресурс]. Режим доступу: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2024 (дата звернення 15.03.2026)

2. Державна служба спеціального зв'язку та захисту інформації України. Аналіз кіберзагроз в Україні за 2024 рік [Електронний ресурс]. Режим доступу: https://cip.gov.ua/ua/news/zvit-shodo-kiberzagroz (дата звернення 15.03.2026)

3. Олдін Л. Кібербезпека: стратегії та тактики. Київ: Видавництво "Фабула", 2021. 352 с.

4. Коваленко О. В., Смірнов О. А. Застосування SIEM-систем для моніторингу та управління подіями інформаційної безпеки в корпоративних мережах// Сучасний захист інформації. 2023. №1(53). С. 22-29.

5. Мельник О. І., Романюк О. Н. Методи виявлення аномалій у мережевому трафіку за допомогою алгоритмів машинного навчання// Матеріали МНПК «Інформаційні технології та безпека». Київ, КПІ, 2025. С. 145-148.

6. IETF. Domain Names - Concepts and Facilities (RFC 1034) [Електронний ресурс]. Режим доступу: https://datatracker.ietf.org/doc/html/rfc1034 (дата звернення 15.03.2026)

7. IETF. Domain Names - Implementation and Specification (RFC 1035) [Електронний ресурс]. Режим доступу: https://datatracker.ietf.org/doc/html/rfc1035 (дата звернення 15.03.2026)

8. Kurose J. F., Ross K. W. Computer Networking: A Top-Down Approach. 8th Edition. Pearson, 2020. 864 p.

9. Таненбаум Е., Везерволл Д. Комп'ютерні мережі. 6-те вид. Київ: Видавництво "Наш Формат", 2020. 944 с.

10. Іванов В. М., Петренко С. О. Дослідження вразливостей протоколу DNS та методи протидії атакам типу Cache Poisoning// Кібербезпека: освіта, наука, техніка. 2024. №4(24). С. 112-120.

11. Zscaler. The State of DNS-Based Threats Report 2024 [Електронний ресурс]. Режим доступу: https://www.zscaler.com/campaign/state-of-dns-threats-report (дата звернення 16.03.2026)

12. Wireshark Foundation. Wireshark User's Guide [Електронний ресурс]. Режим доступу: https://www.wireshark.org/docs/wsug_html_chunked/ (дата звернення 16.03.2026)

13. Orebaugh A., Ramirez G., Beale J. Wireshark & Ethereal Network Protocol Analyzer Toolkit. Syngress, 2006. 448 p.

14. Грищенко В. В., Ткаченко А. М. Використання стеку ELK для моніторингу та аналізу мережевих подій у реальному часі// Вісник Національного університету «Львівська політехніка». Серія: Інформаційні системи та мережі. 2023. №12. С. 56-63.

15. Elastic. Logstash Reference [Електронний ресурс]. Режим доступу: https://www.elastic.co/guide/en/logstash/current/index.html (дата звернення 18.03.2026)

16. Elastic. Elasticsearch Guide [Електронний ресурс]. Режим доступу: https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html (дата звернення 18.03.2026)

17. Elastic. Kibana Guide: Data Views and Dashboards [Електронний ресурс]. Режим доступу: https://www.elastic.co/guide/en/kibana/current/index.html (дата звернення 18.03.2026)

18. Docker. Docker Documentation: Get Started [Електронний ресурс]. Режим доступу: https://docs.docker.com/get-started/ (дата звернення 18.03.2026)

19. Splunk. Splunk Enterprise Overview [Електронний ресурс]. Режим доступу: https://www.splunk.com/en_us/products/splunk-enterprise.html (дата звернення 18.03.2026)

20. Chuvakin A., Schmidt K. Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management. Syngress, 2012. 544 p.

21. Kent K., Souppaya M. Guide to Computer Security Log Management (NIST

SP 800-92). National Institute of Standards and Technology, 2006. DOI: https://doi.org/10.6028/NIST.SP.800-92 (дата звернення 20.03.2026)

22. Oracle. Oracle VM VirtualBox User Manual [Електронний ресурс]. Режим доступу: https://www.virtualbox.org/manual/ (дата звернення 20.03.2026)

23. MikroTik. Cloud Hosted Router (CHR) Manual [Електронний ресурс]. Режим доступу: https://wiki.mikrotik.com/wiki/Manual:CHR (дата звернення 21.03.2026)

24. IETF. The Syslog Protocol (RFC 5424) [Електронний ресурс]. Режим доступу: https://datatracker.ietf.org/doc/html/rfc5424 (дата звернення 21.03.2026)

25. Syslog: протокол мережевого журналювання. Навчальний посібник / О. С. Головня. Житомир: «Житомирська політехніка», 2022. 145 с.

26. MikroTik. Winbox User Guide [Електронний ресурс]. Режим доступу: https://wiki.mikrotik.com/wiki/Manual:Winbox (дата звернення 22.03.2026)

27. MikroTik. Network Address Translation (NAT) [Електронний ресурс]. Режим доступу: https://wiki.mikrotik.com/wiki/Manual:IP/Firewall/NAT (дата звернення 22.03.2026)

28. MikroTik. System Logs and Remote Logging [Електронний ресурс]. Режим доступу: https://wiki.mikrotik.com/wiki/Manual:System/Log (дата звернення 22.03.2026)

29. Docker. Docker Compose documentation [Електронний ресурс]. Режим доступу: https://docs.docker.com/compose/ (дата звернення 23.03.2026)

30. YAML Ain’t Markup Language (YAML™) Version 1.2 [Електронний ресурс]. Режим доступу: https://yaml.org/spec/1.2/spec.html (дата звернення 23.03.2026)

31. Elastic. Logstash Filter Plugins: Grok [Електронний ресурс]. Режим доступу: https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html (дата звернення 24.03.2026)

32. Elastic. Logstash Filter Plugins: Aggregate [Електронний ресурс]. Режим доступу: https://www.elastic.co/guide/en/logstash/current/plugins-filters-aggregate.html (дата звернення 24.03.2026)

33. Elastic. Logstash Filter Plugins: Translate (for domain matching) [Електронний ресурс]. Режим доступу: https://www.elastic.co/guide/en/logstash/current/plugins-filters-translate.html (дата звернення 24.03.2026)

34. Marchal S., Francois J., State R., Engel T. PhishScore: Hacking Phishers' Minds// 10th International Conference on Network and Service Management (CNSM), 2014. DOI: https://doi.org/10.1109/CNSM.2014.7014144 (дата звернення 24.03.2026)

35. Microsoft. Nslookup command reference [Електронний ресурс]. Режим доступу: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/nslookup (дата звернення 25.03.2026)

36. Kali Linux Documentation. Penetration Testing Distribution [Електронний ресурс]. Режим доступу: https://www.kali.org/docs/ (дата звернення 25.03.2026)

37. IETF. Defending against DNS Reflection/Amplification Attacks (RFC 5358) [Електронний ресурс]. Режим доступу: https://datatracker.ietf.org/doc/html/rfc5358 (дата звернення 26.03.2026)

38. CISA. DNS Amplification Attacks | CISA [Електронний ресурс]. Режим доступу: https://www.cisa.gov/news-events/alerts/2013/03/29/dns-amplification-attacks (дата звернення 26.03.2026)

39. Лисенко О. М., Коваленко І. В. Аналіз та моделювання DDoS-атак з використанням DNS-ампліфікації// Системи обробки інформації. 2024. №2(173). С. 45-52.

40. Scapy Documentation. Interactive packet manipulation program [Електронний ресурс]. Режим доступу: https://scapy.readthedocs.io/en/latest/ (дата звернення 28.03.2026)

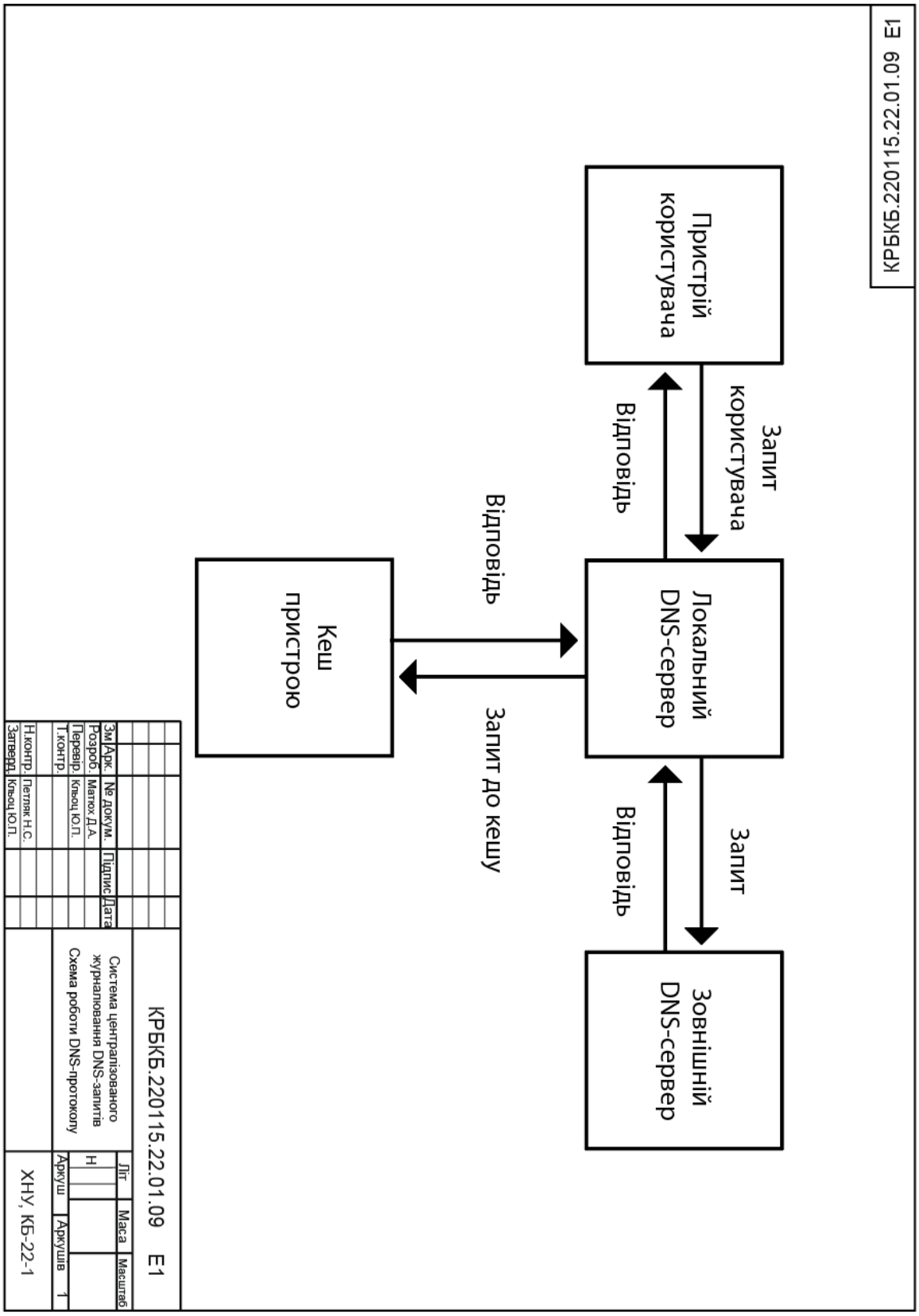
41. IETF. DNS Zone Transfer Protocol (AXFR) (RFC 5936) [Електронний ресурс]. Режим доступу: https://datatracker.ietf.org/doc/html/rfc5936 (дата звернення 30.03.2026)

42. Microsoft. PowerShell Documentation: Invoke-RestMethod [Електронний ресурс]. Режим доступу: https://learn.microsoft.com/en-

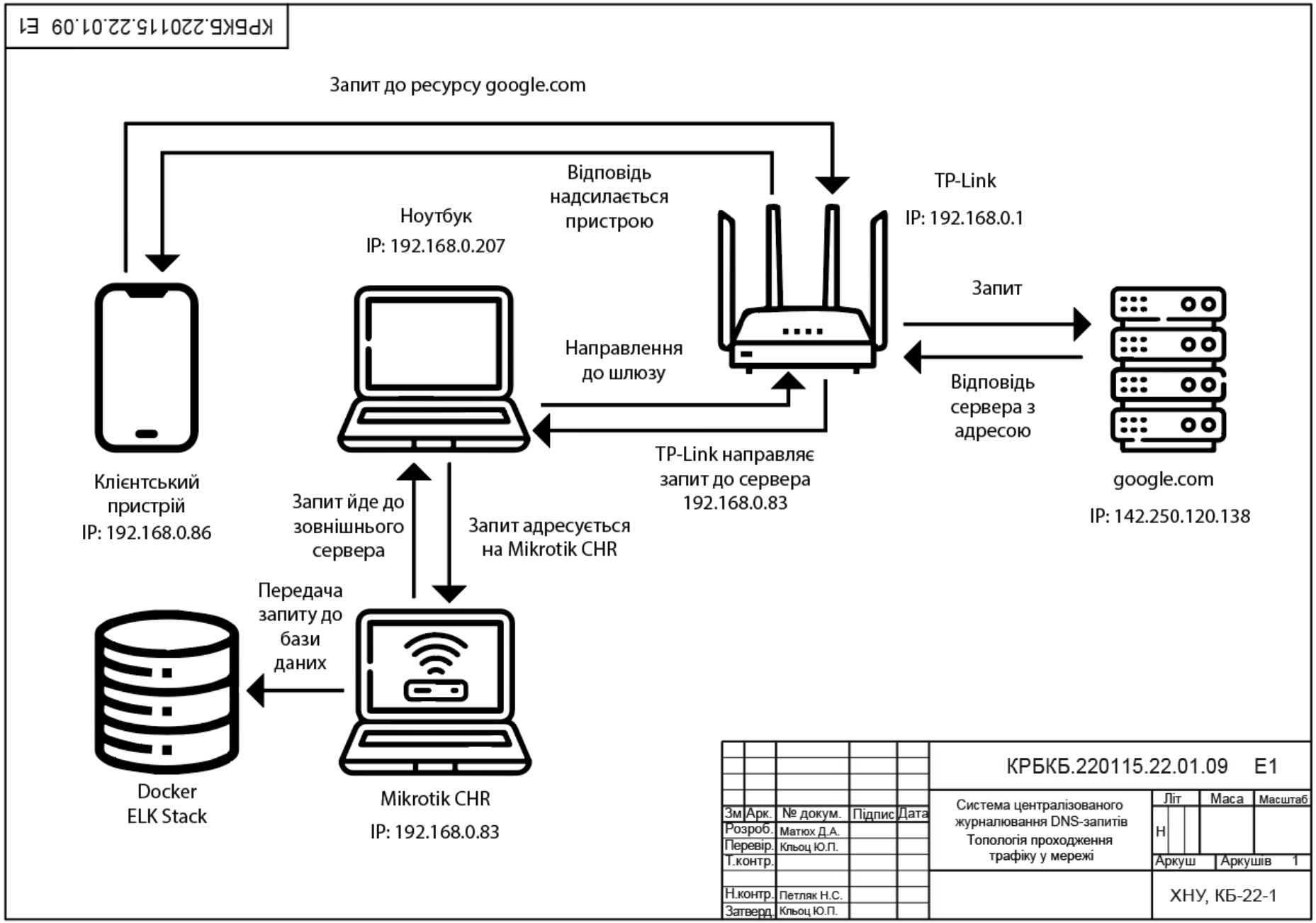
us/powershell/module/microsoft.powershell.utility/invoke-restmethod (дата звернення 02.04.2026)

Додаток А

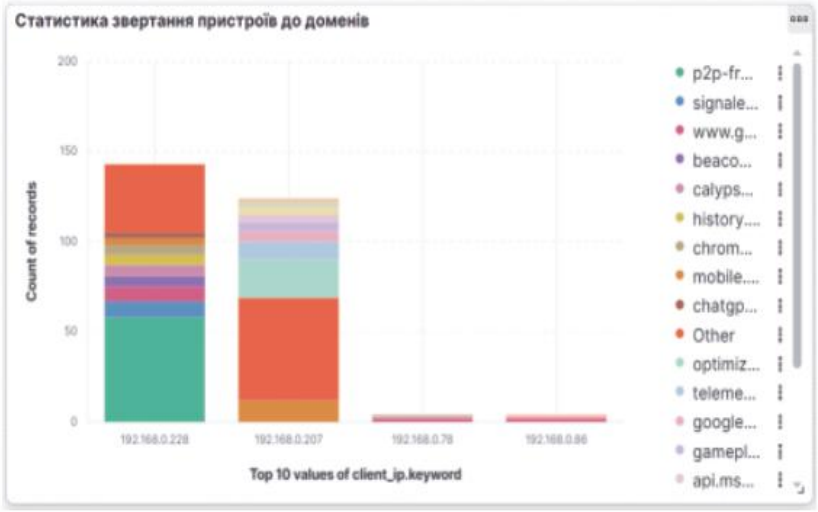
Графічна частина

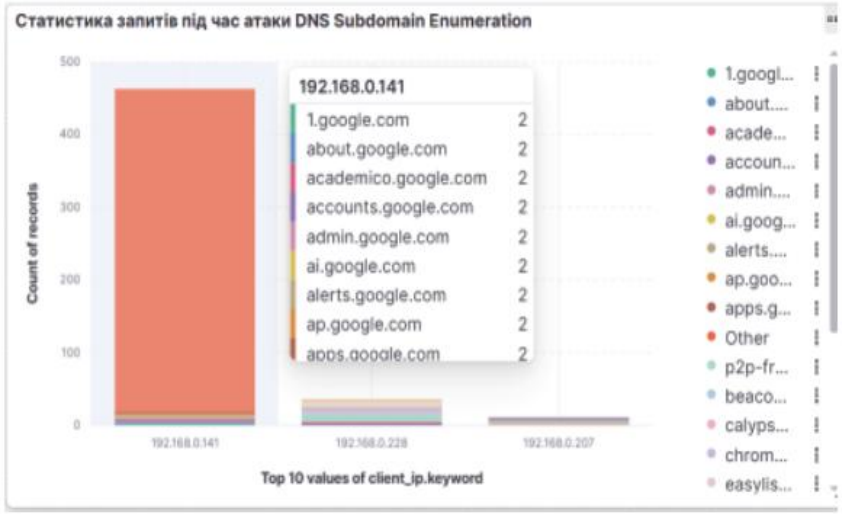


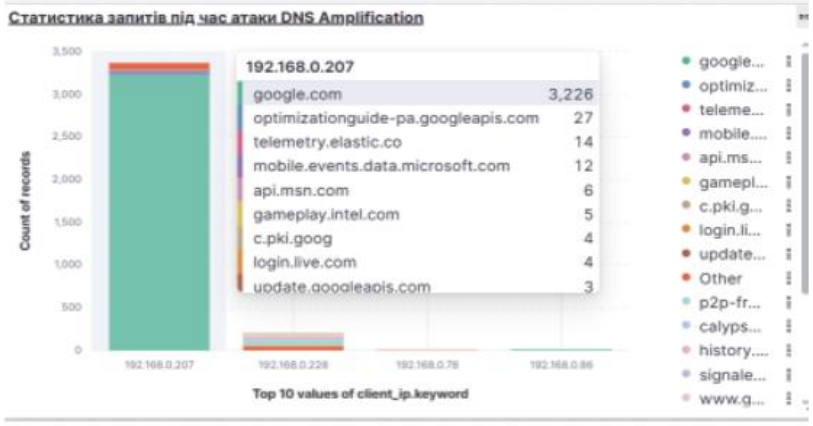
КРБКБ.220115.22.01.09 Е1

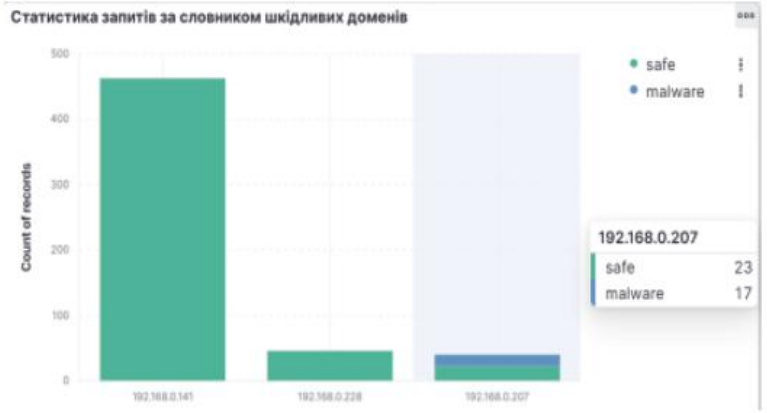


					КРБКБ.220115.22.01.09 Е1	
Зм.	Арк.	№ докум.	Підпис	Дата	Система централізованого журналювання DNS-запитів Топологія проходження трафіку у мережі	Літ / Маса / Масштаб
Розроб.		Матюх Д.А.				Н
Перевір.		Кльоц Ю.П.				Аркуш / Аркушів 1
Т.контр.						
Н.контр.		Петляк Н.С.				ХНУ, КБ-22-1
Затверд.		Кльоц Ю.П.				









					КРБКБ.220115.22.01.09 Е1	
Зм.	Арк.	№ докум.	Підпис	Дата	Система централізованого журналювання DNS-запитів Результати тестування прототипу системи	Літ / Маса / Масштаб
Розроб.		Матюх Д.А.				Н
Перевір.		Кльоц Ю.П.				Аркуш / Аркушів 1
Т.контр.						
Н.контр.		Петляк Н.С.			ХНУ, КБ-22-1	
Затверд.		Кльоц Ю.П.				

Додаток Б

Лістинг коду розробленого прототипу

1) Конфігураційний файл docker-compose.yml

```
services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:8.12.0
    container_name: elasticsearch
    environment:
      - discovery.type=single-node
      - xpack.security.enabled=false
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
    ports:
      - "9200:9200"
    volumes:
      - es_data:/usr/share/elasticsearch/data
    mem_limit: 1g

  logstash:
    image: docker.elastic.co/logstash/logstash:8.12.0
    container_name: logstash
    environment:
      - "LS_JAVA_OPTS=-Xms512m -Xmx512m"
      - pipeline.workers=1
      - pipeline.ordered=true
    volumes:
      - ./logstash.conf:/usr/share/logstash/pipeline/logstash.conf
      - ./dictionaries:/usr/share/logstash/dictionaries
    ports:
      - "514:514/udp"
```

```
    depends_on:
      - elasticsearch
    mem_limit: 2g

  kibana:
    image: docker.elastic.co/kibana/kibana:8.12.0
    container_name: kibana
    ports:
      - "5601:5601"
    environment:
      - ELASTICSEARCH_HOSTS=http://elasticsearch:9200
    depends_on:
      - elasticsearch
    mem_limit: 1g

volumes:
  es_data:
```

2) Конфігураційний файл logstash.conf

```
input {
  udp {
    port => 514
    type => "mikrotik_dns"
  }
}

filter {
  if [type] == "mikrotik_dns" {

    grok {
      match => {
```

```
      "message" => [
        "<%{POSINT}>%{TIMESTAMP_ISO8601:mikrotik_time}
%{DATA} (?<clean_message>dns.*)",
        "<%{POSINT}>%{SYSLOGTIMESTAMP:mikrotik_time}
%{DATA} (?<clean_message>dns.*)"
      ]
     }
     tag_on_failure => []
   }

   if [mikrotik_time] {
     date {
       match => [
         "mikrotik_time",
         "ISO8601",
         "MMM  d HH:mm:ss",
         "MMM dd HH:mm:ss"
       ]
       timezone => "Europe/Kyiv"
       target => "@timestamp"
     }

     mutate {
       replace => { "message" => "%{clean_message}" }
       remove_field => [ "mikrotik_time", "clean_message" ]
     }
   }
   # -----------------------------------

   if [message] =~ /id:/ {
```

```
    grok {
      match => { "message" => "id:%{INT:query_id}" }
      tag_on_failure => []
    }
  } else if [message] =~ /#/ {
    grok {
      match => { "message" => "#%{INT:query_id}" }
      tag_on_failure => []
    }
  }

  grok {
    match => { "message" => [
      "query from %{IP:client_ip}: #%{INT:query_id} %{NOTSPACE:domain}(?:\.)? %{WORD:query_type}",

      "done query: #%{INT:query_id} %{NOTSPACE:domain}(?:\.)? %{IP:resolved_ip}",

      "done query: #%{INT:query_id} %{NOTSPACE:domain}(?:\.)? %{WORD:dns_error_code}",

      "done query: #%{INT:query_id} %{GREEDYDATA:dns_error_message}",

      "got query from %{IP:client_ip}:%{INT:client_port}:",
      "question: %{NOTSPACE:domain}(?:\.)?:%{WORD:query_type}:%{WORD:dns_class}",
```

```
"<%{HOSTNAME:domain}\.(?:\.)?:%{WORD:query_type}:%{INT}=%{IP:resolved_ip}>",
      "--- %{WORD:packet_action} (?:udp query to|reply to|answer from) %{IP:remote_server_ip}:%{INT:remote_server_port}:"
    ]}
    add_tag => [ "dns_parsed" ]
    tag_on_failure => []
  }

  if [query_id] {
    aggregate {
      task_id => "%{query_id}"
      code => "
        if event.get('client_ip')
          map['saved_client_ip'] = event.get('client_ip')
        end
        if map['saved_client_ip']
          event.set('client_ip', map['saved_client_ip'])
        end

        if event.get('client_port')
          map['saved_client_port'] = event.get('client_port')
        end
        if map['saved_client_port']
          event.set('client_port', map['saved_client_port'])
        end
      "
      map_action => "create_or_update"
      end_of_task => false
```

```
    timeout => 10
  }
}

if [domain] {
  mutate {
    gsub => [ "domain", "\.$", "" ]
  }

  translate {
    source => "domain"
    target => "threat_category"
    dictionary_path => "/usr/share/logstash/dictionaries/bad_domains.yml"
    fallback => "safe"
    refresh_interval => 300
  }

  ruby {
    code => "event.set('dns_domain_length', event.get('domain').length)"
  }
}

mutate {
  add_field => { "server_port" => "53" }
}
if [message] =~ /12345/ {
  mutate {
    replace => { "client_port" => "12345" }
  }
}
```

```
  mutate {
    convert => {
      "client_port" => "integer"
      "server_port" => "integer"
      "query_id" => "integer"
      "dns_domain_length" => "integer"
    }
  }


  if [dns_error_code] {
    mutate { add_field => { "dns_response_code" => "%{dns_error_code}"
} }
  } else if [dns_error_message] =~ /NXDOMAIN/ or [message] =~
/NXDOMAIN/ {
    mutate { add_field => { "dns_response_code" => "NXDOMAIN" } }
  } else if [resolved_ip] {
    mutate { add_field => { "dns_response_code" => "NOERROR" } }
  } else if [dns_error_message] {
    mutate { add_field => { "dns_response_code" => "ERROR" } }
  } else {
    mutate { add_field => { "dns_response_code" => "UNKNOWN" } }
  }

  if [dns_response_code] == "NXDOMAIN" {
    mutate {
      add_tag => [ "dns_error", "subdomain_enumeration_suspect" ]
    }
  }
```

```
    if [query_type] == "ANY" or [query_type] == "255" {
      mutate {
        add_tag => [ "dns_amplification_suspect" ]
      }
    }

    if [dns_domain_length] and [dns_domain_length] > 45 {
      mutate {
        add_tag => [ "dns_tunneling_suspect" ]
      }
    }

    mutate {
      remove_field => [ "dns_error_code", "dns_error_message" ]
    }

    if "dns_parsed" not in [tags] and ![query_id] {
      drop { }
    }
  }
}

output {
  elasticsearch {
    hosts => ["http://elasticsearch:9200"]
    index => "dns-logs-%{+YYYY.MM.dd}"
  }
}
```

3) Файл конфігурації скрипта для оновлення списку доменних імен update_blacklist.ps1

```
$url = "https://raw.githubusercontent.com/StevenBlack/hosts/master/data/StevenBlack/hosts"
$outputFile = "dictionaries/bad_domains.yml"

Write-Host "Step 1: Downloading fresh blacklist..."
$data = Invoke-WebRequest -Uri $url -UseBasicParsing

Write-Host "Step 2: Formatting data for Logstash..."
$lines = $data.Content -split "`n" | Where-Object { $_ -match "^0.0.0.0" } | ForEach-Object {
    $domain = ($_ -split "\s+")[1]
    if ($domain -and $domain -ne "0.0.0.0") {
        "`"$domain`": `"malware`""
    }
}
Write-Host "Step 3: Saving to file..."
[System.IO.File]::WriteAllLines((Join-Path (Get-Location) $outputFile), $lines)
Write-Host "Success! Dictionary updated."
```