МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра кібербезпеки

Тимощук Дмитро Іванович

Алгоритми та оцінка функціональності програмного захисту для мінімізації впливу Slowloris атак на вебсервер / Algorithms and evaluation of software protection functionality to minimise the impact of Slowloris attacks on a web server

спеціальність: 125 – Кібербезпека та захист інформації

освітньо-професійна програма – Кібербезпека

Кваліфікаційна робота

Виконав студент групи
КБмз -21
Д.І. Тимощук

Науковий керівник
д.т.н., професор В.В.Яцків

Кваліфікаційну роботу
допущено до захисту:

«____» _____ ___2024 р.

Завідувач кафедри

_____ **В.В.Яцків**

ТЕРНОПІЛЬ - 2024

Факультет комп'ютерних інформаційних технологій
Кафедра кібербезпеки
Освітній ступінь «магістр»
спеціальність: 125 – Кібербезпека та захист інформації
освітньо-професійна програма – Кібербезпека

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ В.В.Яцків
«____» _____ 2023 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

ТИМОЩУКУ Дмитру Івановичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи:
Алгоритми та оцінка функціональності програмного захисту для мінімізації впливу Slowloris атак на вебсервер / Algorithms and evaluation of software protection functionality to minimise the impact of Slowloris attacks on a web server
керівник роботи д.т.н., професор В.В.Яцків

затверджені наказом по університету від 12 грудня 2023 року № 753

2. Строк подання студентом закінченої кваліфікаційної роботи 12 грудня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити:
– провести огляд технології Slowloris атак;
– проаналізувати загальні методи захисту від Slowloris атак;
– налаштувати тестове середовище для моделювання Slowloris атаки на вебсервер;
– удосконалити алгоритм виявлення, блокування та сповіщення про Slowloris атаку;
– розробити програмне забезпечення для виявлення Slowloris атаки та блокування IP-адрес зловмисників.
5. Перелік графічного матеріалу у роботі:
– принцип здійснення Slowloris атаки на вебсервер;

– загальні методи захисту від Slowloris атак;
– архітектура вебсерверів;
– схема тестового середовища ;
– здійснення Slowloris атаки на вебсервер Apache;
– наслідки атаки на вебсервер Apache;
– алгоритм роботи механізму блокування Slowloris атаки;
– інтеграція програмного забезпечення як сервісу;
– блокування IP-адрес атакуючих та логування події;
– повідомлення в Telegram про атаку Slowloris.

6. Консультанти розділів кваліфікаційної роботи

	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 12 грудня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строки виконання етапів кваліфікаційної роботи	Примітка
1	Огляд предметної області	12.2023 р. – 03.2024 р.	
2	Створення та налаштування тестового середовища	03.2024 р. – 06.2024 р.	
3	Розробка та оцінка функціональності захисту	06.2024 р. – 11.2024 р.	

Студент _____ Тимощук Д.І.
(підпис

Керівник роботи _____ д.т.н., професор В.В.Яцків
(підпис)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Алгоритми та оцінка функціональності програмного захисту для мінімізації впливу Slowloris атак на вебсервер» на здобуття освітнього ступеня «Магістр» зі спеціальності 125 «Кібербезпека та захист інформації» освітньо-професійної програми «Кібербезпека» написана обсягом 92 сторінки і містить 38 ілюстрації, 3 додатоки та 35 джерела за переліком посилань.

Метою роботи є підвищення ефективності алгоритмів виявлення Slowloris атак, розробка та оцінка функціональності механізму мінімізації їх впливу на вебсервери Apache під управлінням операційної системи Linux.

Методи досліджень. Для розв’язання поставлених задач у даній кваліфікаційній роботі використано: аналіз загальних методів захисту від Slowloris атак, створення програмного забезпечення з використанням вдосконаленого алгоритму захисту для виявлення Slowloris атаки та блокування IP-адрес зловмисників, проведення експериментальних атак на вебсервер для тестування програмного забезпечення.

Результати дослідження. Удосконалено алгоритм виявлення атаки та створено програмний засіб, який спрямований на захист вебсерверів від Slowloris атак на платформі Linux. Розроблений захист включає в себе інтеграцію з брандмауером для можливості блокування атаки в реальному часі, систему логування атаки та можливість отримання сповіщень на Telegram Bot про інцидент.

Отримані результати свідчать про можливість практичного використання розробленого захисту для підвищення стійкості та безпеки вебсерверів Apache.

Ключові слова: SLOWLORIS, LINUX, APACHE, БРАНДМАУЕР, IPTABLES, TELEGRAM BOT.

ABSTRACT

Qualification work on "Algorithms and evaluation of software protection functionality to minimise the impact of Slowloris attacks on a web server " for the degree of "Master" in the specialty 125 "Cybersecurity and Information Protection" educational and professional program "Cybersecurity" is written in 92 pages and contains 35 illustrations, 3 appendice and 34 source according to the list of links.

The aim of the study is to improve the efficiency of Slowloris attack detection algorithms, develop and evaluate the functionality of a mechanism for minimising their impact on Apache web servers running the Linux operating system.

Research methods. To solve the tasks set in this qualification work, the following methods were used: analysis of general methods of protection against Slowloris attacks, creation of software using an advanced protection algorithm to detect Slowloris attacks and block attackers' IP addresses, conducting experimental attacks on a web server to test the software.

Research results. An attack detection algorithm has been improved and a software tool has been created to protect web servers from Slowloris attacks on the Linux platform. The developed protection includes integration with a firewall to block the attack in real time, an attack logging system, and the ability to receive notifications on Telegram Bot about the incident.

The results obtained indicate the possibility of practical use of the developed protection to improve the resilience and security of Apache web servers.

Keywords: SLOWLORIS, LINUX, APACHE, FIREWALL, IPTABLES, TELEGRAM BOT.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

DOS	—	Denial of Service
HTTP	—	Hypertext Transfer Protocol
URL	—	Uniform Resource Locator
DMZ	—	Demilitarized Zone
NAT	—	Network Address Translation
SIEM	—	Security Information and Event Management

ВСТУП

Актуальність роботи. Інтернет-технології відіграють ключову роль у функціонуванні різноманітних сфер надання послуг Однак, разом із зростанням обсягів онлайн-трафіку, збільшується і загроза для вебсерверів у вигляді кібератак. Однією з таких атак є Slowloris, яка спрямована на заволодіння ресурсами вебсервера та призводить до його відмови. Ця проблема стає дедалі актуальнішою, існуючі засоби захисту від неї не завжди є ефективними та вимагають подальшого вдосконалення. Тому, вдосконалення алгоритмів, розробка та оцінка нового програмного захисту для мінімізації впливу Slowloris атак на вебсервери є важливим завданням у галузі кібербезпеки та актуальною науково-прикладною задачею.

Мета і завдання дослідження. Метою роботи є підвищення ефективності алгоритмів виявлення Slowloris атак, розробка та оцінка функціональності механізму мінімізації їх впливу на вебсервери Apache під управлінням операційної системи Linux.

Досягнення визначеної мети передбачає вирішення таких завдань:

– провести огляд методики здійснення Slowloris атаки;

– проаналізувати загальні методи захисту від Slowloris атак;

– налаштувати тестове середовище для моделювання Slowloris атаки на вебсервер;

– удосконалити алгоритм та розробити програмне забезпечення для виявлення Slowloris атаки та блокування IP-адрес зловмисників;

– реалізувати можливості запуску програмного забезпечення як сервісу для постійної роботи на вебсервері;

– розробити механізм запису інформації про виявлені атаки в лог-файл для подальшого аналізу;

– розробити механізм сповіщення в Telegram Bot про здійснення атаки;

Об’єкт дослідження. Об'єктом дослідження є вебсервери, що функціонують під управлінням операційної системи Linux, зокрема

вебсервер Apache, які піддаються атакам типу Slowloris.

Предмет дослідження. Предметом дослідження є методи та алгоритми виявлення і мінімізації впливу Slowloris атак на вебсервери, включаючи розробку програмного забезпечення для захисту вебсерверів від таких атак.

Методи досліджень. Для розв'язання поставлених задач у даній кваліфікаційній роботі використано: аналіз загальних методів захисту від Slowloris атак, створення програмного забезпечення з використанням вдосконаленого алгоритму захисту для виявлення Slowloris атаки та блокування IP-адрес зловмисників, проведення експериментальних атак на вебсервер для тестування програмного забезпечення.

Наукова новизна одержаних результатів. Наукова новизна одержаних результатів кваліфікаційної роботи полягає в удосконаленні алгоритму та практичній реалізації програмного захисту, який спрямований на захист вебсерверів від Slowloris атак на платформі Linux. Розроблений захист включає в себе інтеграцію з брандмауером для можливості блокування атаки в реальному часі, систему логування атаки та можливість отримання сповіщень на Telegram Bot про інцидент.

Практичне значення отриманих результатів. Практичне значення одержаних результатів полягає в можливості використання розробленого програмного захисту для підвищення стійкості вебсерверів до Slowloris атак. Використання цього захисту може допомогти адміністраторам та спеціалістам з кібербезпеки покращати надійності та безпеку вебсервісів.

Публікації та апробація КР.

1. Tymoshchuk D., Yatskiv V. Using hypervisors to create a cyber polygon. Measuring and computing devices in technological processes. 2024. No. 3. P. 52–56. URL: https://doi.org/10.31891/2219-9365-2024-79-7

2. Tymoshchuk D., Yatskiv V. Slowloris DDoS detection and prevention in real-time. Theoretical and empirical scientific research: concept and trends. 2024. URL: https://doi.org/10.36074/logos-16.08.2024.036.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основні Поняття Slowloris атаки

Slowloris атака - це вид атаки на вебсервер, спрямований на заволодіння його ресурсами і відмову у обслуговуванні легітимних користувачів [1]. Назва "Slowloris" походить від назви тварини - лориса (англ. "loris"). Лорис - це малий нічний примат, який досить повільний та майстерно пересувається через гілки дерев. Аналогія полягає в тому, що Slowloris атака дуже повільна і спокійно розтягується в часі, спостерігаючи за своєю жертвою і споживаючи її ресурси поступово, як лорис переміщується по гілках.

Основні поняття, пов'язані з Slowloris атакою, включають кілька важливих моментів. По-перше, сервер – це вебсервер, який обробляє запити від клієнтів, що є атакуючими. Клієнт, у свою чергу, є програмою, що запитує в вебсервера інформацію та використовує Slowloris техніку для здійснення атаки. Slowloris техніка полягає в тому, що атакуючий намагається утримати велику кількість відкритих з'єднань з вебсервером, що призводить до виділення сервером ресурсів на кожне відкрите з'єднання [2].

Це викликає недоступність вебсервера для легітимних користувачів, оскільки всі доступні ресурси спрямовані на обслуговування атакуючих. Одним з ключових моментів Slowloris атаки є затримка запитів, коли атакуючий намагається затримати відправку повних запитів, відкривши з'єднання, але не завершуючи їх повністю. Це змушує сервер очікувати завершення запиту та утримувати відкриті з'єднання.

Slowloris працює, використовуючи заголовок «Keep-Alive» запиту HTTP. Цей заголовок інформує сервер про те, що в майбутньому будуть додаткові запити, тому зловмисник може надіслати кілька незавершених запитів і тримати їх відкритими [3]. Це призводить до того, що сервер досягає ліміту з’єднань і не може обробляти нові запити від законних користувачів. Потім зловмисник надсилає додаткові неповні запити, щоб

зберегти контроль над існуючими з’єднаннями та запобігти доступу законних користувачів до сервера. Надсилаючи достатню кількість даних з кожним запитом, але не завершуючи його, зловмисники ефективно зв’язують ресурси веб-сервера на тривалий період часу (рисунок 1.1).

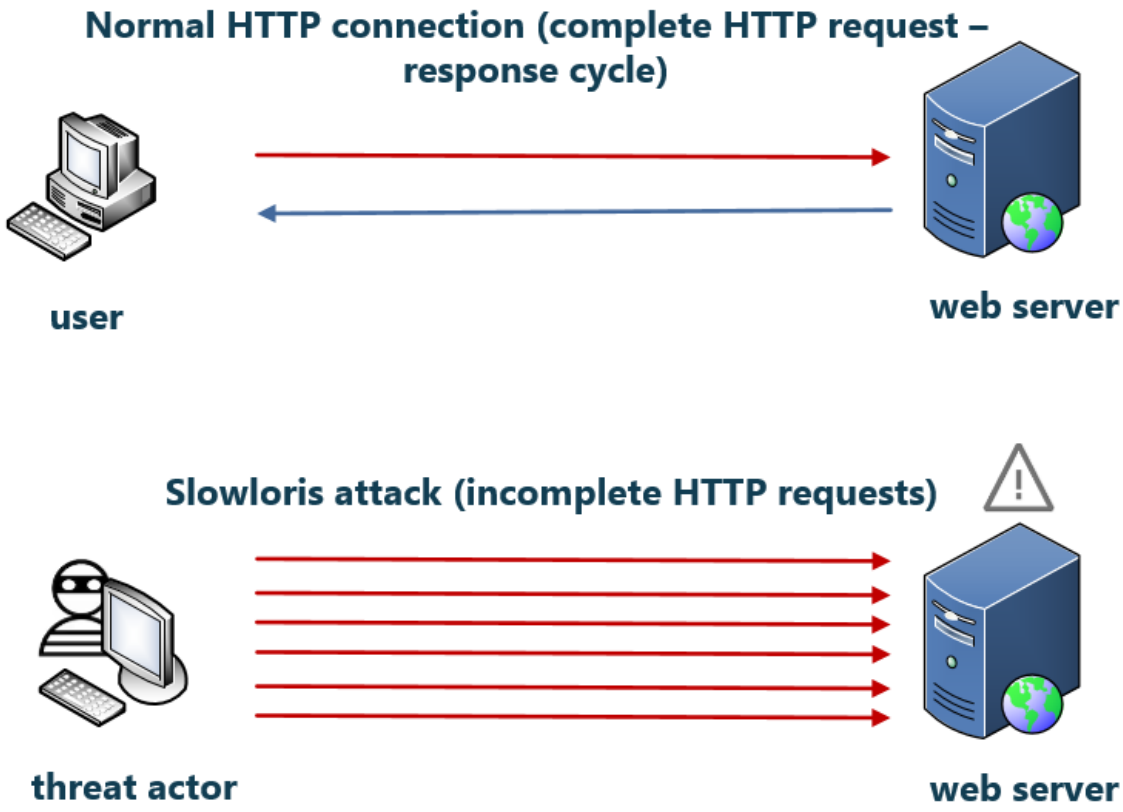

Рисунок 1.1 –Принцип здійснення Slowloris атаки на вебсервер

Зловмисники можуть використовувати бот мережі або інші мережеві ресурси для одночасного надсилання багатьох запитів на вебсервер (рисунок 1.2). Це дозволяє збільшити силу атаки і її здатність перевантажити сервер [4]. Атакуючий поступово виснажує ресурси сервера, і це може бути надзвичайно ефективним при великій кількості одночасних з'єднань.

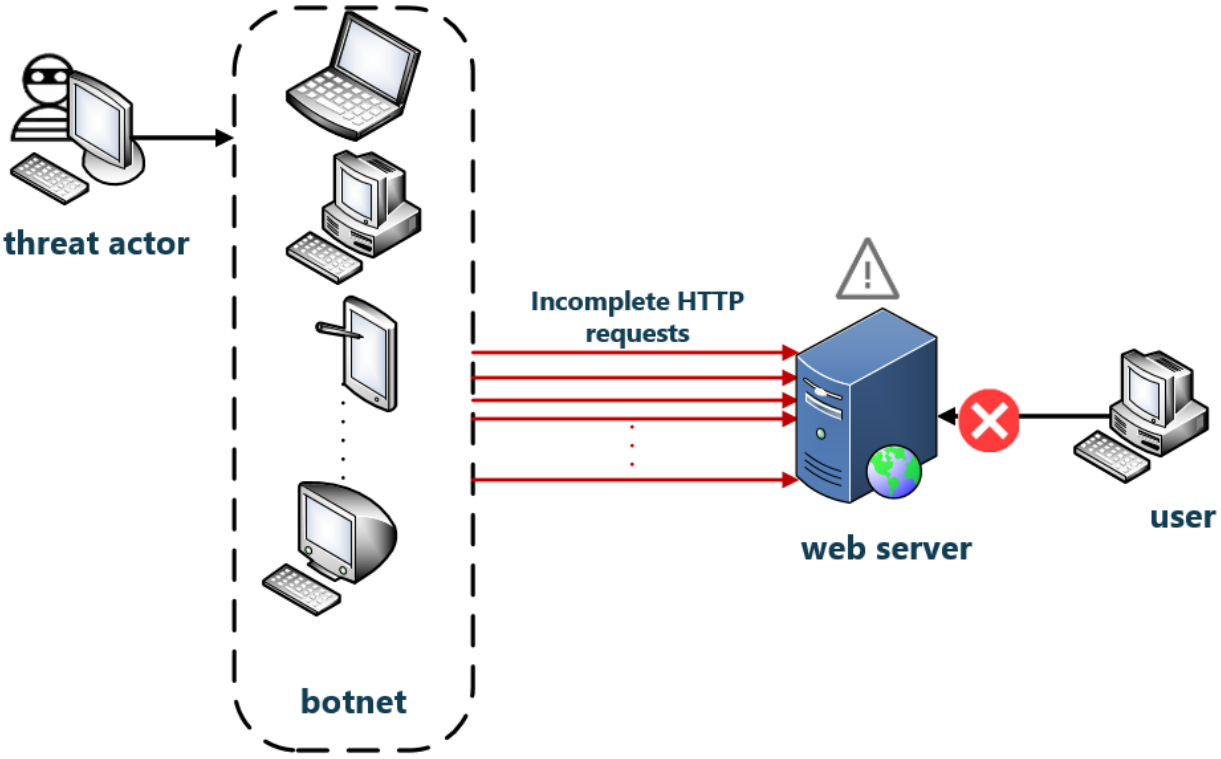

Рисунок 1.2 –Принци здійснення Slowloris атаки з використанням бот мережі

Використання пристроїв IoT для Slowloris атаки є особливо небезпечним, враховуючи зростаючу кількість таких пристроїв та їхню обмежену безпеку. Пристрої IoT зазвичай мають обмежені обчислювальні ресурси та прості налаштування безпеки, що робить їх вразливими до взлому та подальшого використання в ботнетах для здійснення атак. Атакуючі можуть захопити контроль над численними IoT-пристроями, об’єднуючи їх у бот мережі. Потім ця бот мережа може використовуватися для здійснення Slowloris атак на вебсервери. Завдяки великій кількості пристроїв, атака може бути надзвичайно ефективною, навіть якщо кожен окремий пристрій надсилає лише невелику кількість запитів.

Оскільки Slowloris атака характеризується низьким обсягом трафіку, що відправляється атакуючими це робить її важкою для виявлення, оскільки вона не генерує значний обсяг даних.

1.2 Цілі та результати Slowloris атаки

Однією з основних цілей Slowloris атаки є заволодіння ресурсами вебсервера. Атакуючий намагається використовувати всі доступні з'єднання до сервера, перевантажуючи його ресурси та споживаючи їх

Один з очевидних результатів Slowloris атаки - це тимчасова або тривала недоступність вебсервера для легітимних користувачів. Сервер може бути перевантажений і не здатний відповідати на запити. Це може вплинути на користувачів та призвести до втрати прибутку для власників вебсервера.

Slowloris атака може призвести до того, що сервер реагує повільно на запити. Це може вплинути на користувачів, які спостерігають за зменшенням швидкості обробки їхніх запитів.

Slowloris атака має серйозний вплив на вебсервери та їхні користувачі. Вона спрямована на досягнення різних цілей, включаючи заволодіння ресурсами сервера, вимоги викупу, створення недоступності та вплив на репутацію. Захист від Slowloris атаки є важливим для забезпечення безпеки вебсерверів. Ця атака може призвести до значних втрат для бізнесу, який обслуговується вебсервером, і може завдати шкоду репутації організації.

1.3 Огляд загальних методів захисту від Slowloris атак

Захисту від Slowloris атак включає в себе різноманітні техніки та підходи, спрямовані на зменшення ризику та мінімізацію наслідків цієї атаки.

Один з перших кроків у захисті від Slowloris - це вдосконалення конфігурації вебсервера, зокрема збільшення максимальної кількості одночасних з'єднань, яку сервер може обробляти. Також, важливо налаштувати таймаути для завершення неактивних з'єднань.

Параметр що вказує максимальну кількість одночасних з'єднань, які вебсервер може обробляти може значно покращити стійкість до цієї форми атаки. Зазвичай встановлюється значно вище, ніж за замовчуванням, оскільки однією з основних характеристик Slowloris атаки є велика кількість

одночасних незавершених з'єднань. Налаштування цього параметра на значення, яке вище від типового навантаження, може зробити вебсервер менш вразливим до атак. Але потрібно враховувати що можливості сервера не є безмежні і також можуть бути вичерпані.

Параметр, що визначає час очікування для завершення активного з'єднання після отримання запиту може зменшити шкоду від атаки. Зменшення цього значення допомагає зменшити час, протягом якого атакуючий може утримувати активне з'єднання без відправки повного запиту. Зазвичай рекомендується налаштовувати це значення на низький рівень, наприклад, 5-10 секунд. Але при використанні великої бот-мережі з великою кількістю одночасних запитів ефект від застосування даного параметру буде зменшуватись.

Вебсервери мають модулі або плагіни, які розроблені з врахуванням попередніх параметрів для зменшення впливу Slowloris атаки.

При здійсненні атаки з багатьох джерел використання модулів для Apache, які можуть пом’якшити атаку не дає бажаного ефекту [5]. Оскільки mod_reqtimeout ефективно відкидає повільні з’єднання, але це не перешкоджає створенню цих з’єднань. Це означає, що під час пільгового періоду очікування mod_reqtimeout слоти підключення Apache можуть бути заповнені. mod_antiloris запобігає надто великій кількості повільних з'єднань, створених з одного клієнта. Це усуває недолік mod_reqtimeout, але при використанні бот мережі це також не приносить очікуваного ефекту. mod_limitipconn концептуально схожий на mod_antiloris, але mod_limitipconn спрацьовує занадто пізно і не може пом’якшити Slowloris. Він призначений для повернення HTTP 503 клієнту, якщо клієнт робить занадто багато з’єднань, але припускає, що клієнт бажає отримати відповідь. Slowloris, звичайно, цього не потребує, тому слоти підключення залишаються зайнятими.

Реверс-проксі сервер може виступати в якості посередника між клієнтами та вебсервером, розподіляючи та обмежуючи трафік [6]. Цей посередник приймає всі запити від клієнтів та передає їх вебсерверу.

Реверс-проксі сервер може розподіляти навантаження між вебсерверами (рисунок 1.3).

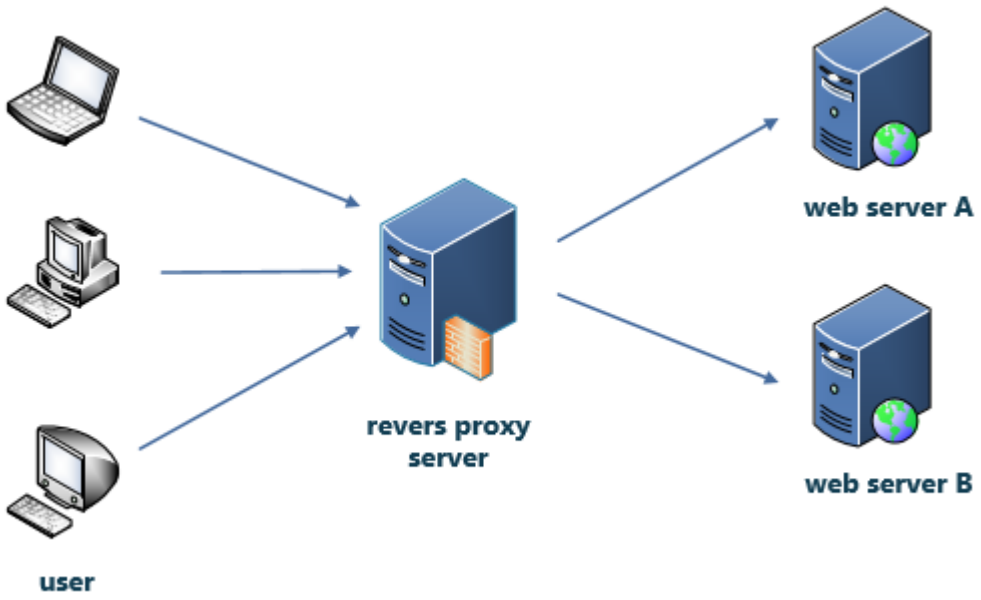

Рисунок 1.3 –Принци роботи реверс-проксі сервера

Це означає, що він може розділити запити між кількома екземплярами вебсервера або серверами за допомогою балансування навантаження. Якщо один із серверів стає мішенню Slowloris атаки, інші сервери все одно можуть обслуговувати запити, зменшуючи вплив атаки. Реверс-проксі сервер може аналізувати всі вхідні запити перед тим, як передати їх вебсерверу. Він може використовувати різні правила та фільтри, щоб виявляти аномалії, які можуть бути характерні для Slowloris атаки. Проксі може встановлювати таймаути для з'єднань та обмеження на кількість одночасних з'єднань, які можуть бути створені з одного IP-адреси. Це допомагає зменшити час, протягом якого Slowloris атака може утримувати активне з'єднання та зменшити можливість створення багатьох незавершених з'єднань.

За допомогою реверс-проксі сервера можна створити додатковий бар'єр між клієнтами та вебсервером, що зменшує вразливість сервера до Slowloris атаки. Крім того, реверс-проксі може вдосконалити загальну безпеку та ефективність сервера, забезпечуючи захист від інших загроз.

Іншим методом зменшення впливу атаки є використання брандмауерів для блокування IP-адрес, з яких здійснюються Slowloris атаки. Розробка алгоритму, методики цього процесу та його автоматизація може допомогти в ефективному виявленні та блокуванні атакуючих.

Брандмауер (firewall) - це програма або апаратний пристрій, який контролює та фільтрує мережевий трафік на основі правил та політик безпеки [7].

Один з основних методів захисту від Slowloris атаки - це блокування IP-адрес, з яких здійснюються атаки. Брандмауер може бути налаштований для обмеження кількості одночасних з'єднань, які можуть бути встановлені з одного IP-адреси. Це допомагає запобігти перевантаженню вебсервера Slowloris атакою. Але при використанні великої бот-мережі з великою кількістю одночасних запитів з різних IP адрес ефект від застосування цієї можливості буде зменшуватись.

Важливо регулярно оновлювати правила та політики брандмауера, оскільки атакуючі можуть адаптувати свої методи. Брандмауери є важливим компонентом безпеки для захисту вебсерверів від Slowloris атак та інших загроз. Налаштований правильно брандмауер допомагає зменшити вплив атаки та забезпечити безпеку сервера та даних користувачів.

Моніторинг та логування є важливою частиною захисту вебсерверів від Slowloris атаки та інших загроз. Цей процес включає в себе запис подій та активності сервера для подальшого аналізу та виявлення підозрілої або аномальної активності. Вебсервери зазвичай мають журнали подій, які фіксують важливі події, такі як HTTP-запити, відповіді сервера, помилки та інше. Моніторинг та аналіз цих журналів допомагає виявляти підозрілу активність, таку як надмірну кількість з'єднань від однієї IP-адреси, що може

бути ознакою Slowloris атаки. Моніторинг може включати в себе автоматизований аналіз журналів для виявлення аномальної активності. Наприклад, система може виявити надмірну кількість спроб з'єднання з однієї IP-адреси та сповістити адміністратора про це.

Моніторинг може бути інтегрований з автоматизованими системами реагування. Важливо вести докладні журнали подій активності сервера. Журнали мають включати в себе важливу інформацію, таку як IP-адреси клієнтів, час подій, URL-адреси запитів, відповіді сервера та інше. Це допоможе в аналізі атак та ідентифікації підозрілих IP-адрес.

Журнали повинні регулярно аналізуватися адміністраторами безпеки. Це дозволяє вчасно виявляти загрози та вживати заходів для їх виявлення та запобігання. Журнали подій містять важливу інформацію, і важливо забезпечити їх захист від несанкціонованого доступу. Доступ до журналів повинен бути обмежений та захищений від витоку інформації.

Моніторинг та логування важливі для виявлення Slowloris атак та інших загроз, а також для реагування на них. Цей процес допомагає підтримувати безпеку вебсервера та користувачів, а також сприяє вчасному реагуванню на потенційні загрози.

Вдосконалення загальної безпеки сервера є також важливим аспектом захисту від Slowloris атаки та інших загроз. Загальна безпека сервера включає в себе різні заходи та налаштування, що забезпечують його стійкість до різних видів атак і небажаних втручань. Регулярне оновлення операційної системи, вебсервера та інших компонентів є критичним для вдосконалення безпеки сервера. Виробники випускають патчі, які виправляють виявлені вразливості, тому важливо слідкувати за оновленнями та встановлювати їх якомога швидше.

Також важливим є мінімізація поверхні атаки (Attack Surface Reduction). Деякі компоненти та служби сервера можуть бути відключені, якщо вони не використовуються, для зменшення атакованої поверхні. Наприклад, непотрібні служби або модулі можуть бути відключені.

Доступ до сервера та його ресурсів повинен бути обмежений. Використання принципу найменших прав (Principle of Least Privilege) допомагає забезпечити те, що користувачі та процеси будуть мати лише необхідні права доступу.

Регулярне резервне копіювання важливих даних і налаштувань сервера дозволяє відновлювати сервер у випадку атаки або непередбачених проблем.

Важливо також приділити увагу безпеці на рівні додатків, використовуючи засоби захисту, які вбудовані у додатки або фреймворки.

Якщо сервер обслуговується командою адміністраторів, важливо, щоб вони були освічені в галузі безпеки та здатні розпізнавати підозрілу активність.

Вдосконалення загальної безпеки сервера вимагає системного та комплексного підходу до захисту. Це допомагає забезпечити надійну роботу сервера та зменшити його вразливість до Slowloris атаки та інших загроз.

1.4 Вебсервер як ціль атаки

1.4.1 Огляд роботи вебсерверів

Загальний принцип роботи вебсерверів - це процес надання веб-сторінок та інших ресурсів клієнтам через протокол HTTP (HTTPS). Вебсервери є основними компонентами, які забезпечують доступ до вебсторінок у всесвітній мережі [8].

Вебсервер очікує на запити від клієнтів, які можуть бути веббраузерами, мобільними додатками або іншими програмами. Запити надходять через мережу, зазвичай за допомогою протоколу HTTP (TCP порт 80) або HTTPS (TCP порт 443). Кожен запит містить URL, який ідентифікує ресурс, який клієнт бажає отримати. Вебсервер аналізує URL та визначає, який ресурс клієнт запитує. Це може бути HTML-сторінка, зображення, відео, аудіофайл, сценарій, статичний файл або динамічно генерований вміст.

Залежно від типу ресурсу, вебсервер може обробляти запит різними способами. Якщо запитується статичний файл, такий як зображення або HTML-сторінка, сервер може просто надіслати файл клієнту без змін. У випадку динамічно генерованого контенту, сервер виконує відповідний сценарій (PHP, Python або інший) для створення вмісту перед відправкою клієнту.

Після обробки запиту вебсервер генерує відповідь для клієнта. Відповідь включає в себе HTTP-статусний код (наприклад, 200 OK для успішного запиту або 404 Not Found для відсутності ресурсу) та заголовки, які містять інформацію про вміст відповіді. Вміст може бути HTML-кодом, текстом, зображеннями, відео, аудіо або будь-яким іншим видом даних.

Після генерації відповіді сервер відправляє її клієнту через мережу. Відповідь міститься у вигляді HTTP-пакету та передається через вебпротокол клієнту.

Вебсервери можуть обслуговувати багато запитів одночасно від різних клієнтів, що забезпечує швидкий доступ до вмісту для багатьох користувачів одночасно.

Після відправки відповіді вебсервер завершує з'єднання з клієнтом. Зазвичай, це виконується за допомогою механізму закриття з'єднання в протоколі HTTP.

1.4.2 Архітектура вебсерверів

Вебсервери можна розділити на два основних типи архітектури: потокові та асинхронні [9].

Apache є потоковим вебсервером [10]. Потокові вебсервери використовують модель обробки запитів, де кожен запит обробляється окремим потоком або процесом.

Коли клієнт надсилає запит до потокового вебсервера, сервер створює окремий потік або процес для обробки цього запиту. Це означає, що кожен

запит обробляється відокремлено від інших запитів [11]. У потокових серверах, кожен потік або процес блокується під час обробки запиту. Це може призвести до блокування всього сервера в разі великої кількості запитів. Потокові сервери часто дозволяють розробникам використовувати синхронний код, що полегшує розробку додатків. Розробникам не потрібно дбати про обробку конкурентних процесів або потоків. Багато потокових серверів, зокрема Apache, підтримують розширення та модулі, які можна використовувати для розширення функціональності сервера. Apache обслуговує статичний вміст, використовуючи традиційний файловий підхід. Однією з переваг вебсервера Apache є його здатність внутрішньо обробляти динамічний вміст, не покладаючись на зовнішні компоненти. Він обробляє динамічний вміст, інтегруючи процесор відповідних мов програмування в кожен робочий екземпляр. Користувачі можуть активувати цей процесор за допомогою динамічно завантажуваних модулів. Apache дозволяє користувачам встановлювати понад 50 офіційних і сторонніх модулів. Він також підтримує динамічне завантаження модулів для більш ефективного використання пам'яті. Хоча основні функції сервера завжди доступні, користувачі можуть завантажувати та вивантажувати модулі, щоб змінювати їх. Модулі Apache можуть виконувати різні завдання, такі як обробка динамічного вмісту, встановлення змінних середовища та переписування URL-адрес.

Вебсервери Apache підтримують додаткову конфігурацію кожного каталогу за допомогою файлів .htaccess. Вони дозволяють непривілейованим користувачам контролювати певні аспекти вебсайту, не надаючи їм доступу до редагування основних конфігураційних файлів.

Nginx - це приклад асинхронного сервера [12]. Асинхронні вебсервери використовують модель обробки запитів, де один потік або процес може обслуговувати багато запитів без блокування. Nginx використовує підхід із асинхронною, неблокуючою, керованою подіями архітектурою [13]. Це дозволяє вебсерверу обробляти кілька підключень в рамках одного процесу.

Nginx має головний процес, який виконує привілейовані операції, такі як прив’язка до портів, читання і оцінка файлів конфігурації та створення дочірніх процесів.

Асинхронні сервери розроблені так, щоб один потік або процес міг обслуговувати багато запитів одночасно, без блокування. Це досягається за допомогою асинхронних операцій та нечіткого обмеження ресурсів. Завдяки асинхронній архітектурі, сервер витрачає менше ресурсів на створення та керування потоками або процесами, що дозволяє йому бути більш продуктивним та швидким. Асинхронні сервери використовують неблокуючу модель обробки, що дозволяє серверу обробляти запити без очікування завершення попередніх операцій. Nginx ідеально підходять для обслуговування багатьох одночасних з'єднань, що робить їх популярними для задач розподілу навантаження та обслуговування великих обсягів трафіку.

Nginx також відомий своєю здатністю працювати як реверс-проксі сервер та здійснювати кешування, що полегшує розгортання складних інфраструктур [14]. З точки зору обслуговування статичного вмісту, Nginx швидше, ніж Apache, оскільки він кешує статичні файли.

Однак Nginx не має вбудованої можливості динамічної обробки вмісту. Для обробки динамічного вмісту він повинен передати запити зовнішньому процесору, наприклад FastCGI Process Manager (PHP-FPM). Цей зовнішній процесор інтерпретує запити в динамічний вміст і надсилає результат назад на вебсервер. Коли вебсервер отримає вміст, він передасть результати клієнту. Nginx пропонує різні офіційні та сторонні модулі для інтеграції в основне програмне забезпечення. Однак він не підтримує динамічне завантаження модулів. Їх потрібно скомпілювати у рамках основного програмного забезпечення.

На відміну від Apache, Nginx не підтримує налаштування на рівні каталогу. Хоча це призводить до обмеженої гнучкості, це допомагає підвищити ефективність сайту. Оскільки Nginx немає функції пошуку та

інтерпретування файлів .htaccess це дозволяє обслуговувати запити швидше ніж Apache. Nginx забезпечує безпеку сервера, забороняючи додаткову конфігурацію, оскільки лише користувачі з правами root можуть змінювати налаштування сервера та сайту.

Обираючи між потоковим і асинхронним сервером, варто враховувати конкретні потреби проекту, ресурси сервера та очікувану завантаженість. Велика частина веб-сайтів та додатків використовують обидва типи серверів у своїх архітектурах.

2 СТВОРЕННЯ ТА НАЛАШТУВАННЯ ТЕСТОВОГО СЕРЕДОВИЩА

2.1 Схема тестового середовища

На рисунку 2.1 показана схема тестового середовища для моделювання Slowloris атаки.

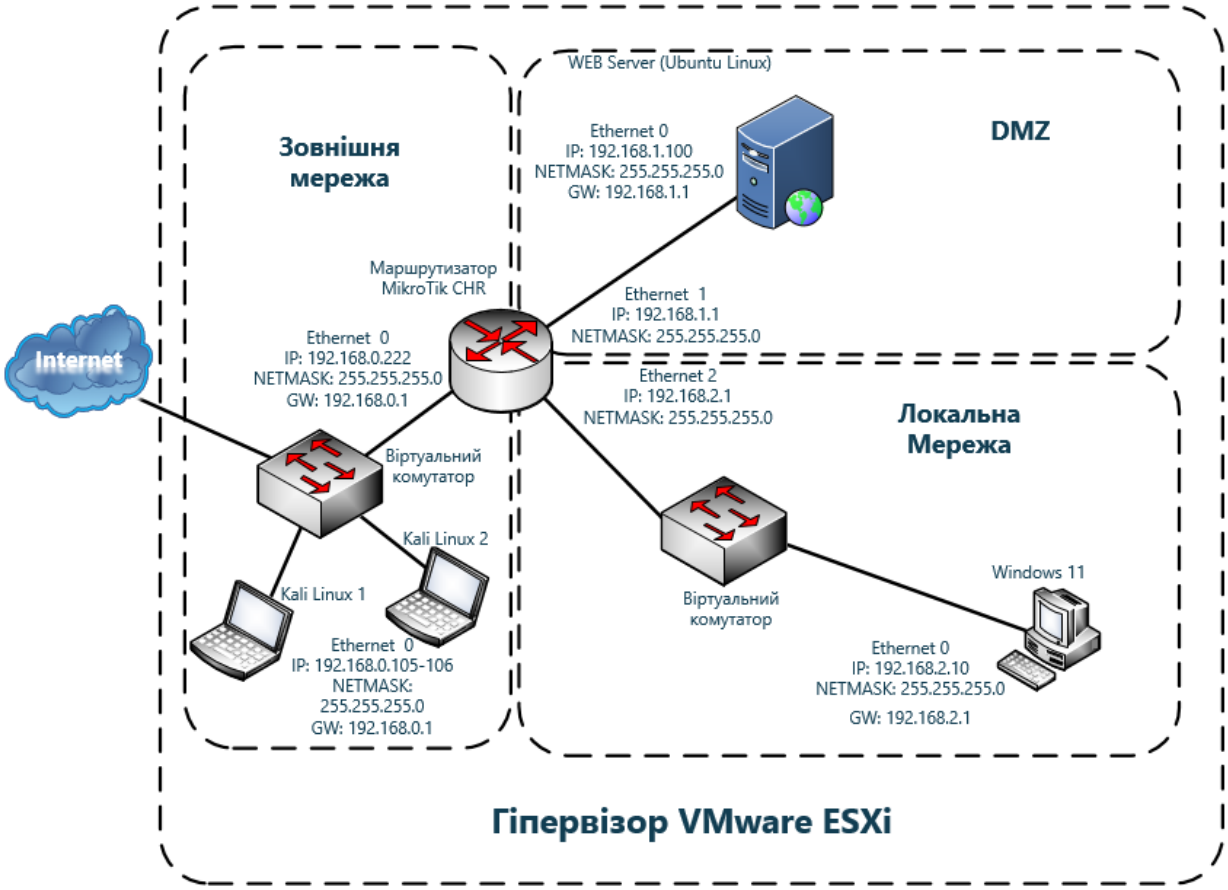

Рисунок 2.1 – Схема тестового середовища для проведення Slowloris атаки

Схема тестового середовища для проведення Slowloris атаки на вебсервер складається з наступних елементів: гіпервізор VMware ESXi, зовнішня мережа, DMZ, внутрішня мережа, маршрутизатор MikroTik CHR.

Гіпервізор VMware ESXi є основою схеми та відповідає за віртуалізацію ресурсів фізичного сервера і управління віртуальними машинами, які на ньому запущені. ESXi використовується для розділення одного фізичного сервера на кілька віртуальних машин, кожна з яких має

власні ОС. Це основа для створення ізольованих та контрольованих тестових середовищ.

Зовнішня мережа представляє собою середовище, звідки атака Slowloris може бути ініційована. Зовнішня мережа підключена до Інтернету через віртуальний комутатор. В цій мережі є дві віртуальні машини з Kali Linux, які використовуються для здійснення атак та тестування безпеки. Їх IP-адреси 192.168.105.105 та 192.168.105.106, маска підмережі 255.255.255.0 та шлюз 192.168.0.1. Ці машини будуть використані як вихідні точки для ініціації Slowloris атаки.

DMZ - це сегмент мережі, який використовується для ізоляції публічно доступних серверів від основної частини корпоративної мережі, щоб у випадку компрометації цих серверів потенційний зловмисник не міг безпосередньо атакувати внутрішні ресурси компанії. В DMZ зазвичай розміщують сервери, які мають бути доступними з Інтернету, такі як вебсервери, сервери електронної пошти або DNS сервери. В сегменті DMZ знаходиться вебсервер Apache на базі операційної системи Ubuntu Linux. Вебсервер має IP-адресу 192.168.1.100, маску підмережі 255.255.255.0 та шлюз 192.168.1.1. Цей сервер є основним цільовим об'єктом для атаки Slowloris.

Локальна мережа містить ВМ Windows 11, яка буде використана для тестування доступності вебсервера до, підчас та після Slowloris атаки. Windows 11 має IP-адресу 192.168.2.10, маску підмережі 255.255.255.0 та шлюз 192.168.2.1.

Маршрутизатор MikroTik CHR є віртуальним пристроєм, що працює на гіпервізорі VMware ESXi і виконує кілька важливих функцій в мережі. MikroTik CHR керує всім мережевим трафіком між зовнішньою мережею та іншими сегментами, такими як DMZ та локальна мережа. Це включає направлення трафіку між віртуальними машинами та інтернетом. Він діє як брандмауер, обмежуючи доступ до внутрішніх ресурсів з Інтернет та між різними сегментами мережі. Цей маршрутизатор є ключовим елементом для

забезпечення безпеки, управління доступом та ефективної маршрутизації трафіку в тестовому середовищі для випробування Slowloris атак. Його віртуальна природа дозволяє швидко адаптувати мережеву структуру до змінюваних вимог безпеки та тестування.

Мережеві налаштування наступні:

- Ethernet 0 (зовнішні мережа) з IP-адресою 192.168.0.222, маскою підмережі 255.255.255.0 та шлюзом 192.168.0.1;
- Ethernet 1 (мережа DNZ) з I IP-адресою 192.168.1.1, маскою підмережі 255.255.255.0;
- Ethernet 2 (локальна мережа) з I IP-адресою 192.168.2.1, маскою підмережі 255.255.255.0.

Створена схема тестового середовища допоможе навчитися розпізнавати, захищати та реагувати на атаки Slowloris у контрольованих лабораторних умовах.

2.2 Огляд гіпервізора VMware ESXi

Використання віртуалізації на основі гіпервізора VMware ESXi для створення тестового середовища для моделювання Slowloris атаки є ефективним способом ізоляції та тестування атак у контрольованому середовищі.

VMware ESXi - це надійний гіпервізор типу 1, що дозволяє віртуалізувати сервери, забезпечуючи ефективне використання апаратних ресурсів [15]. ESXi відомий своєю стабільністю, високою продуктивністю та безпекою, що робить його одним із найпопулярніших рішень для віртуалізації [16].

ESXi працює безпосередньо на фізичному обладнанні, замінюючи традиційну операційну систему (рисунок 2.2).

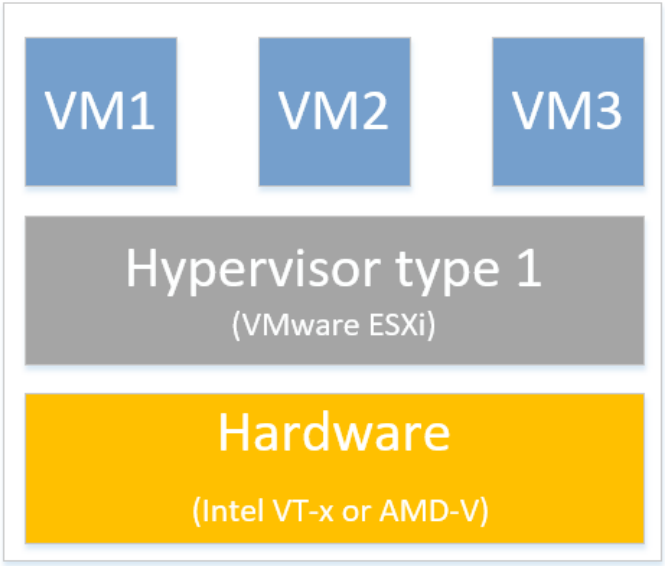

Рисунок 2.2 – Архітектура гіпервізора типу 1

Це дозволяє мінімізувати накладні витрати на віртуалізацію та підвищити продуктивність віртуальних машин. Малий розмір ядра гіпервізора також сприяє підвищенню надійності системи.

За допомогою інструментів управління ESXi можна легко налаштовувати параметри віртуальних машин, такі як кількість виділених процесорних ядер, оперативної пам'яті та мережевих ресурсів, для створення різних сценаріїв атаки та тестування реакції вебсервера.

Технології Intel VT-x і AMD-V є апаратними розширеннями, які призначені для підтримки віртуалізації на рівні процесора. Вони дозволяють гіпервізорам, таким як VMware ESXi ефективніше використовувати апаратні ресурси для запуску віртуальних машин [17].

Intel VT-x (Intel Virtualization Technology) - це технологія, розроблена Intel для полегшення віртуалізації на процесорах цієї компанії [18]. Вона дозволяє гіпервізору безпосередньо керувати і контролювати ресурсами процесора, ізолюючи роботу віртуальних машин від базової операційної системи та інших ВМ. VT-x дозволяє гіпервізору працювати з підвищеною продуктивністю, оскільки зменшує накладні витрати на емуляцію апаратного забезпечення.

AMD-V (AMD Virtualization Technology) - аналогічна технологія від AMD, яка забезпечує підтримку віртуалізації на процесорах цієї компанії [19]. Як і VT-x, вона дозволяє віртуальним машинам працювати більш ефективно, знижуючи накладні витрати на віртуалізацію та підвищуючи продуктивність ВМ.

Обидві технології надають функціональні можливості, які дозволяють гіпервізорам виконувати операції в режимах, що безпосередньо підтримуються на рівні процесора. Це включає можливість гнучкого розподілу ресурсів між віртуальними машинами, ізоляцію ВМ для підвищення безпеки, а також підтримку механізмів швидкого перемикання між гостьовими операційними системами.

Використання технологій Intel VT-x і AMD-V є обов'язковим для забезпечення повної підтримки сучасних віртуалізаційних рішень, що робить їх важливими для розгортання продуктивних і безпечних віртуальних середовищ. В лабораторному середовищі використано обладнання з процесором Intel з підтримкою VT-x.

Для створення такого тестового середовища було встановлено VMware ESXi на фізичний сервер. Після цього через вебінтерфейс ESXi створились необхідні віртуальні машини, що виконуватимуть різні ролі в тестовій мережі (рисунок 2.3).

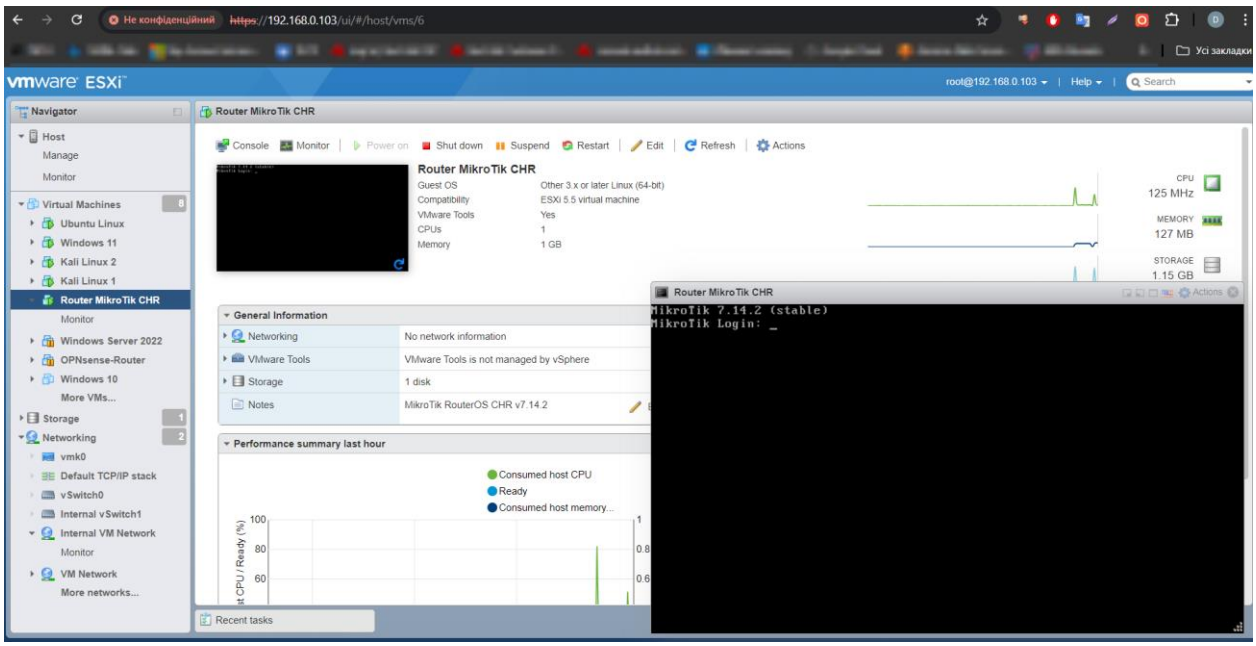

Рисунок 2.3 – Інтерфейс управління VMware ESXi

Віртуалізація на основі VMware ESXi надає можливість ізолювати процес тестування від основної інфраструктури, забезпечуючи безпеку та цілісність основної мережі. Крім того, ESXi дозволяє швидко створювати знімки (snapshots) віртуальних машин, що полегшує відновлення середовища до початкового стану після завершення тестів. Це забезпечує можливість проведення багаторазових експериментів з різними налаштуваннями без потреби в постійному відновленні всієї системи вручну.

2.3 Огляд маршрутизатор MikroTik CHR

MikroTik CHR (Cloud Hosted Router) - це віртуальний маршрутизатор, який працює на різних платформах віртуалізації.. MikroTik CHR може працювати на будь-якій платформі віртуалізації, таких як VMware, Hyper-V, XEN, KVM та інші [20]. Також він підтримує контейнери, що дозволяє легко масштабувати мережеві рішення. Даний маршрутизатор використано в тестовому середовищі на основі гіпервізора VMware ESXi для створення середовища моделювання Slowloris атаки.

Висока продуктивність MikroTik CHR досягається завдяки використанню віртуальних ресурсів сервера, таких як процесор, оперативна пам'ять, мережеві плати та швидкісні дискові накопичувачі. MikroTik CHR підтримує багатоядерні процесори та великий об'єм оперативної пам'яті для забезпечення високої продуктивності.

RouterOS - це операційна система, розроблена компанією MikroTik, яка використовується для маршрутизаторів та мережевого обладнання. Вона забезпечує широкий спектр функцій, що дозволяють ефективно управляти мережевою інфраструктурою, включаючи маршрутизацію, брандмауер, VPN, та управління мережею. RouterOS включає в себе розширені функції безпеки, такі як фільтри брандмауера, шифрування VPN з'єднань за допомогою IPsec, SSTP, L2TP, та інші методи. Вона також надає інструменти для управління якістю обслуговування (QoS), що дозволяє пріоритизувати трафік і забезпечувати стабільну роботу критичних додатків [21].

Цей віртуальний маршрутизатор підтримує всі функції операційної системи RouterOS та має розширені можливості управління мережею через графічний інтерфейс (WinBox), командний рядок (CLI) або вебінтерфейс.

На рисунку 2.4 показано інтерфейс WinBox для управління MikroTik RouterOS на віртуальному маршрутизаторі CHR.

Цей інтерфейс надає адміністратору можливість налаштовувати та контролювати всі аспекти роботи маршрутизатора, від конфігурації мережевих інтерфейсів до управління маршрутизацією та безпекою мережі.

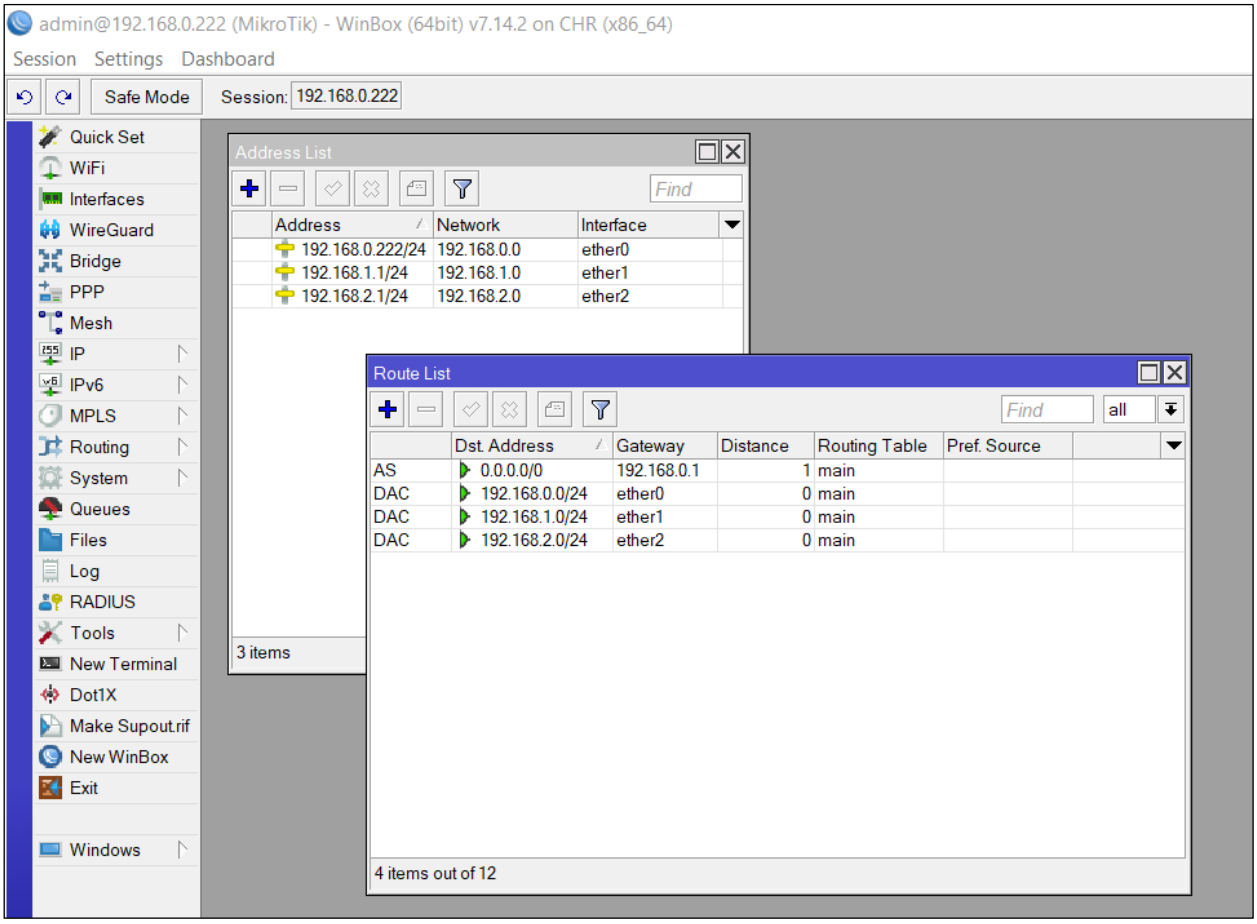

Рисунок 2.4 – Інтерфейс управління віртуального маршрутизатора MikroTik CHR

Маршрутизатор MikroTik CHR підтримує архітектуру x86 64-біт. Версія пакету RouterOS має бути v6.34 або новіша. Центральний процесор (CPU) хосту повинен бути 64-бітним з підтримкою віртуалізації. Рекомендована оперативна пам'ять (RAM) становить 1024 МБ або більше, при цьому мінімальна кількість оперативної пам'яті – 128 МБ, а максимальна – 128 ГБ. Мінімальний розмір диску для віртуального жорсткого диска CHR складає 128 МБ, при цьому максимальний розмір – 16 ГБ.

MikroTik CHR дозволяє легко масштабувати мережу завдяки можливості додавання нових віртуальних екземплярів на тих самих фізичних серверах або у хмарних середовищах. Підтримка різних хмарних платформ, таких як AWS, Azure, Google Cloud, забезпечує гнучкість у виборі середовища розгортання.

2.4 Вебсервер Apache на базі операційна система Ubuntu Linux.

Ubuntu - це популярний дистрибутив операційної системи Linux, який базується на Debian Linux [22]. Він є вільним та відкритим програмним забезпеченням і надає користувачам можливість використовувати операційну систему безкоштовно.

Ubuntu відомий своєю простотою встановлення та використання, багатофункціональністю та підтримкою великої кількості програм та пакунків. Він часто використовується як операційна система для серверів, а також як операційна система робочого столу для особистих комп'ютерів [23].

Ubuntu має регулярні випуски з оновленнями, що забезпечують актуальні версії ядра Linux і програмного забезпечення.

Операційна система Ubuntu надає зручне середовище для розгортання та управління вебсерверами. Зазвичай на Ubuntu встановлюють вебсервери, такі як Apache та Nginx, та налаштовують їх для обробки запитів від клієнтів та надання вебсторінок або інших ресурсів через протокол HTTP

На рисунку 2.5 показано параметри налаштування мережі в Ubuntu Linux.

```
root@ubuntu2204ws:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG    100    0        0 ens33
[redacted]
192.168.1.0     0.0.0.0         255.255.255.0   U     100    0        0 ens33
root@ubuntu2204ws:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:e7:26:86 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.1.100/24 brd 192.168.1.255 scope global noprefixroute ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::8bf6:2998:87ba:c88/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Рисунок 2.5 – Параметри налаштування мережі в Ubuntu Linux

У нашій тестовій схемі ми в якості вебсервера для атаки будемо використовувати Apache.

Apache це відкритий вебсерверний програмний комплекс, який використовується для обслуговування вебсторінок та інших вебресурсів через протокол HTTP. Apache був розроблений і вперше випущений в 1995 році.

Apache є вільним та відкритим програмним забезпеченням, що розповсюджується під ліцензією Apache License. Це означає, що ви можете безкоштовно використовувати та модифікувати Apache відповідно до вашого власного використання. Він володіє великою модульністю, що дозволяє додавати функціональність до сервера за допомогою модулів. Існує багато розширень та модулів для Apache, які дозволяють обробляти PHP, Perl, Python, SSL, компресію даних і багато іншого. Apache підтримується на різних операційних системах, включаючи Linux, Windows, macOS, та Unix.

Apache відомий своєю стабільністю та безпекою. Активна спільнота регулярно випускає оновлення для виправлення виявлених вразливостей.

Він дозволяє налаштовувати правила доступу і автентифікації, що дозволяє обмежувати доступ до веб-ресурсів лише для авторизованих користувачів.

Apache досить продуктивний та може витримувати великий навантаження, особливо коли він правильно налаштований та оптимізований. Також Apache підтримує віртуальні хости, що дозволяє обслуговувати кілька вебсайтів на одному сервері. Apache є одним з найпопулярніших вебсерверів у світі та широко використовується для розгортання вебсайтів та вебдодатків. Багато великих та малих організацій використовують Apache як свій основний вебсервер через його надійність.

Пори всі свої переваги Apache в своїй потоковій архітектурі має вразливість до атак Slowloris [9]. Веб сервер Nginx має іншу архітектуру (дивись 2.1.2) та при умові відповідної конфігурації не схильний до вразливості атак типу Slowloris [12].

З виводу команди `systemctl status apache2` можна переконатись Apache запущено як сервіс в Ubuntu Linux (рисунок 2.6).

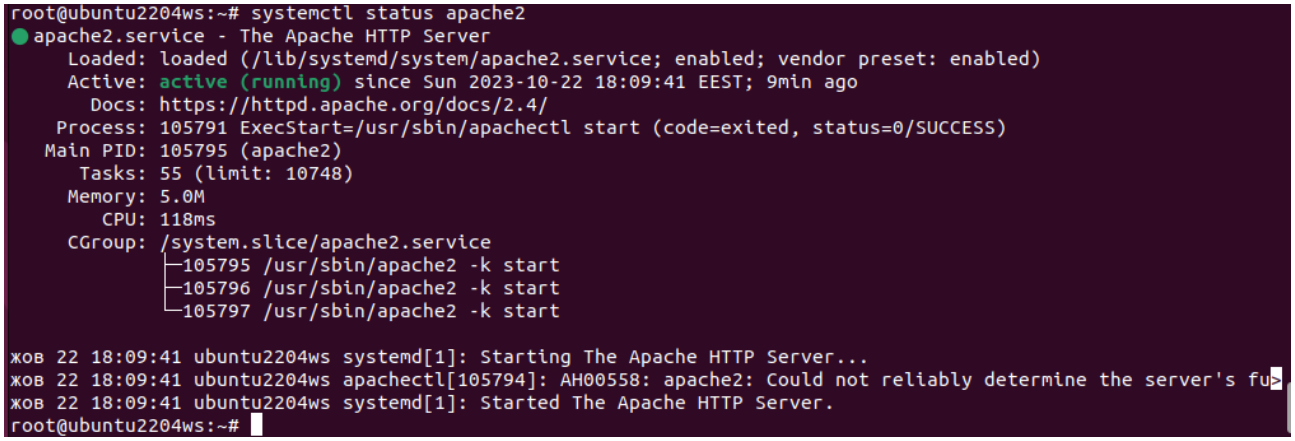

Рисунок 2.6 –Вивід команди `systemctl status apache2` в Ubuntu

Отже вебсервер налаштовано та запущено для потреб тестування.

2.5 Операційна система Kali linux як засіб атаки

Kali Linux - це спеціалізований дистрибутив Linux, розроблений для здійснення тестування на проникнення (penetration testing), відкритого дослідження та аудиту безпеки мереж і комп'ютерів [24]. Ця операційна система зазвичай використовується професіоналами з області інформаційної безпеки, етичними хакерами та аудиторами безпеки для оцінки вразливостей та захищеності інформаційних систем. Kali Linux поставляється з великою кількістю засобів для тестування на проникнення, таких як Metasploit, Hydra і багато інших [25]. Ці інструменти допомагають виявляти вразливості та проводити атаки на системи з метою зміцнення їх безпеки. Також включає інструменти для сканування мережі та виявлення вразливостей у програмному забезпеченні, операційних системах та конфігураціях серверів. Кілька інструментів в Kali Linux дозволяють використовувати виявлені вразливості для здійснення атак, включаючи атаки на віддалені комп'ютери, вебсайти та додатки. Засоби для аналізу мережевого трафіку, такі як Wireshark, допомагають аналізувати та перехоплювати пакети даних для розуміння мережевих комунікацій.

Kali Linux також включає інструменти для проведення атаки на соціальну інженерію, які допомагають ввести в оману користувачів та отримувати доступ до систем.

На рисунку 2.7 показано налаштування мережі в Kali Linux.

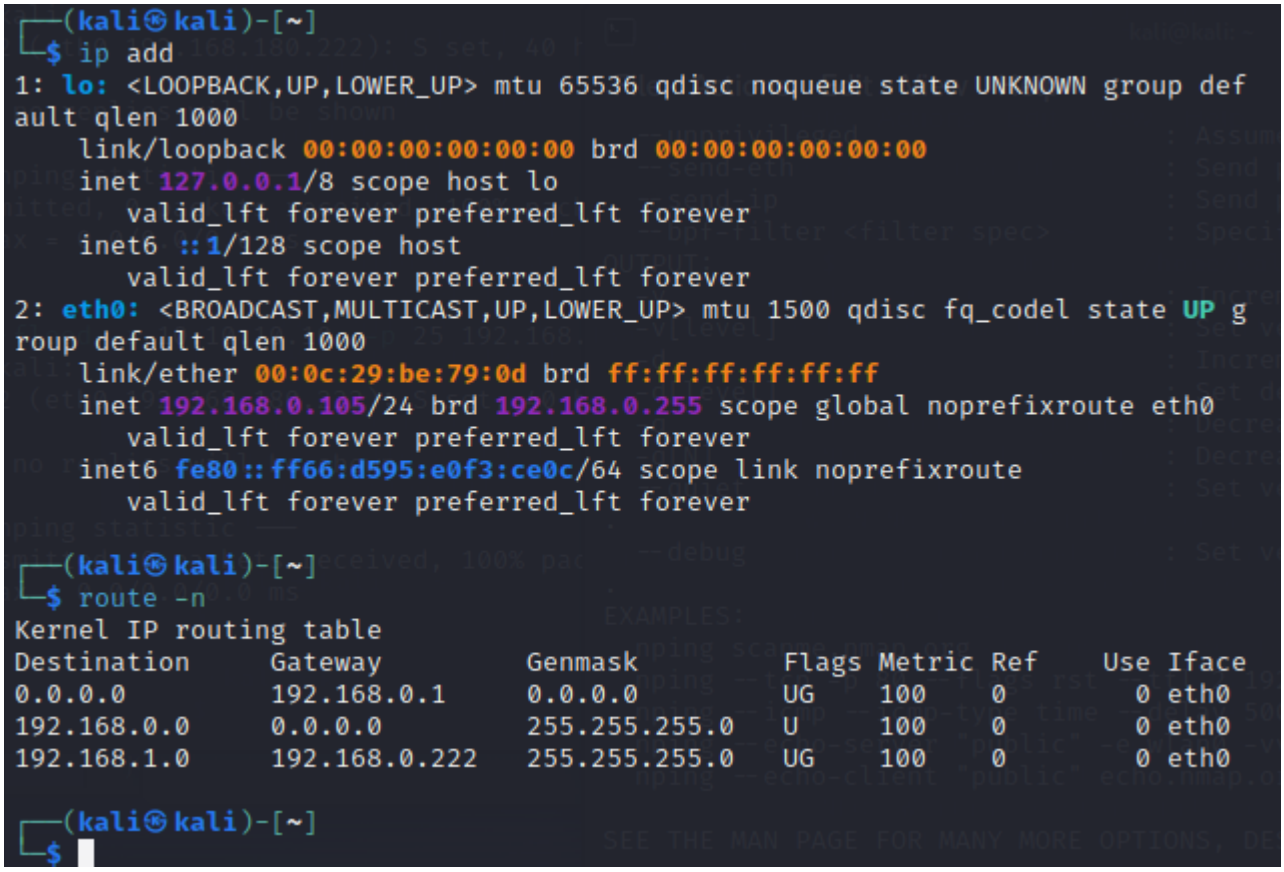

Рисунок 2.7 – Мережеві налаштування операційної системи Kali Linux

На двох операційних з Kali linux буде використано metasploit для здійснення атаки Slowloris. Metasploit - це популярний фреймворк для тестування на проникнення, який містить багато інструментів для виконання атак і тестування безпеки систем [26]. Модуль auxiliary/dos/http/slowloris дає можливість виконати атаку Slowloris на вебсервер [27].

2.6 Здійснення Slowloris атаки на вебсервер

Slowloris - це вид атаки DoS на вебсервер, який спрямований на вичерпання ресурсів сервера та створення завад для доступу до веб-сайту

користувачів [28]. Під час атаки Slowloris зловмисник надсилає багато HTTP-запитів на цільовий вебсервер, але на відміну від звичайної DOS або DDOS атаки, запити надсилаються повільно протягом тривалого періоду часу. Також HTTP-запити неповні, щоб зберегти з'єднання відкритими якомога довше.

На першому етапі тестування здійснимо перевірку працездатності вебсервера з операційної системи Windows 11, яка розміщена в локальній мережі (див. рисунок.2.1).

На рисунку 2.8 показано мережеві налаштування операційної системи Windows 11.

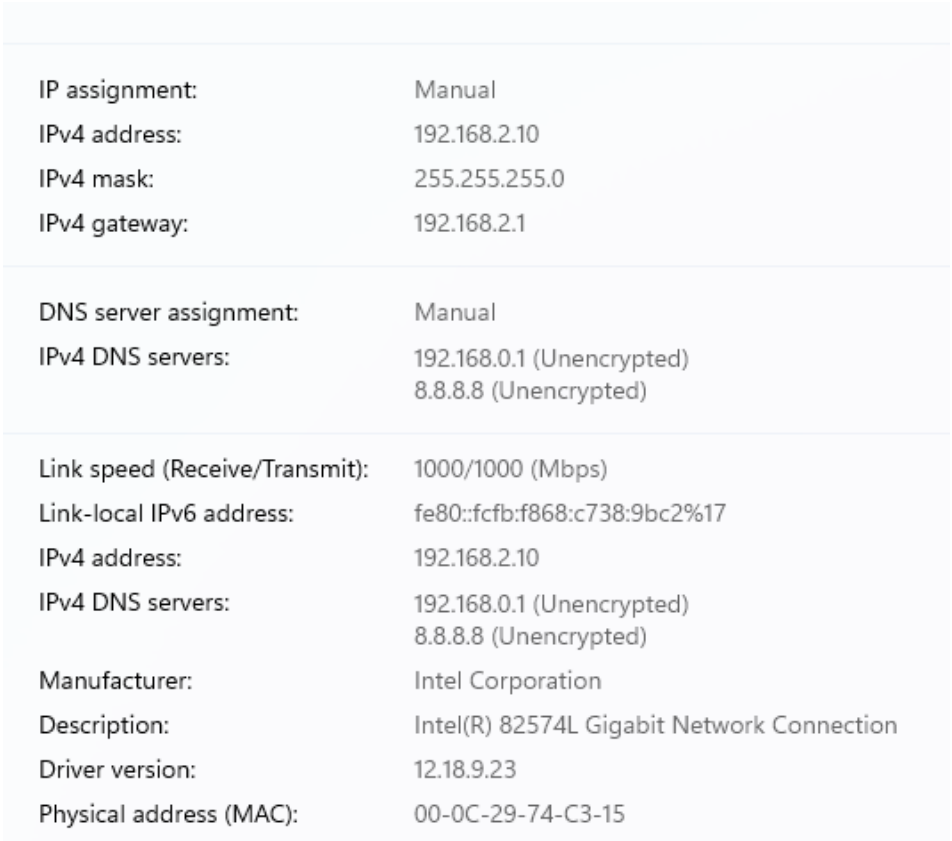

Рисунок 2.8 – Мережеві налаштування операційної системи Windows 11

На рисунку 2.9 показано перевірку доступності вебсервера. Тестовий сайт, який розміщений на вебсервері завантажився швидко та повністю.

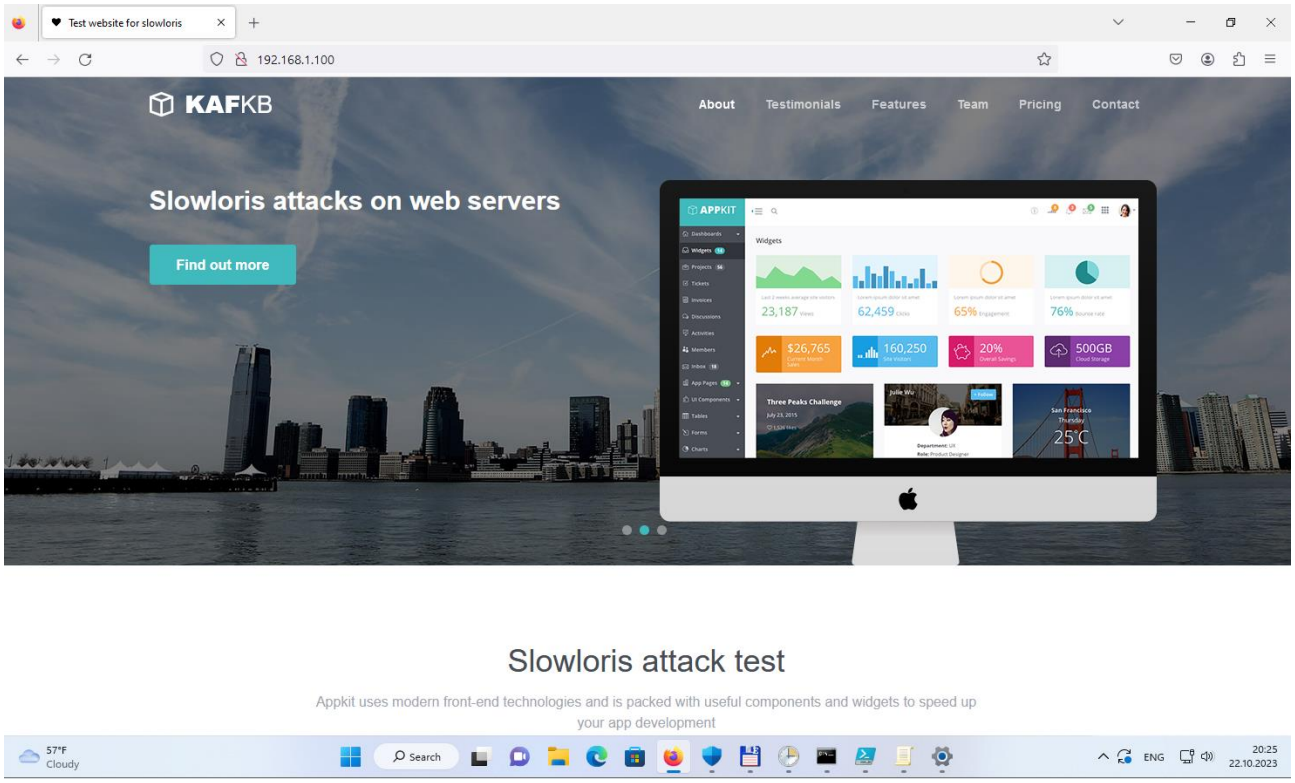

Рисунок 2.9 – Завантаження тестового сайту

Також з виводу команди `tcpdump` [29] на вебсервері можна побачити що клієнт надіслав коректний запит до сервера (рисунок 2.10).

```
10:57:35.328030 IP (tos 0x0, ttl 127, id 1767, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.2.10.59371 > 192.168.1.100.80: Flags [S], cksum 0x889d (correct), seq 3293755956, win 6424
0, options [mss 1460,nop,wscale 8,nop,nop,sackOK], length 0
10:57:35.328077 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.1.100.80 > 192.168.2.10.59371: Flags [S.], cksum 0x84e5 (incorrect -> 0x24c5), seq 1683488
624, ack 3293755957, win 64240, options [mss 1460,nop,nop,sackOK,nop,wscale 7], length 0
10:57:35.329757 IP (tos 0x0, ttl 127, id 1768, offset 0, flags [DF], proto TCP (6), length 40)
    192.168.2.10.59371 > 192.168.1.100.80: Flags [.], cksum 0x5c86 (correct), seq 1, ack 1, win 1026,
length 0
10:57:35.330272 IP (tos 0x0, ttl 127, id 1769, offset 0, flags [DF], proto TCP (6), length 479)
    192.168.2.10.59371 > 192.168.1.100.80: Flags [P.], cksum 0x5a3c (correct), seq 1:440, ack 1, win 1
026, length 439: HTTP, length: 439
        GET / HTTP/1.1
        Host: 192.168.1.100
        User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0
        Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
        Accept-Language: en-US,en;q=0.5
        Accept-Encoding: gzip, deflate
        Connection: keep-alive
        Upgrade-Insecure-Requests: 1
        If-Modified-Since: Thu, 19 Oct 2023 19:42:24 GMT
        If-None-Match: "6de2-60816f4842356-gzip"
```

Рисунок 2.10 – Коректний запит до вебсервера

– Цей HTTP-запит був відправлений до вебсервера за допомогою методу GET. Основні дані включають:

– `GET / HTTP/1.1` - це початкова стрічка запиту, яка містить метод запиту (GET), URI запиту ("/"), та версію протоколу HTTP (HTTP/1.1);

– `Host: 192.168.1.100` - це заголовок, який вказує на IP-адресу вебсервера, до якого надсилається запит;

– `Connection: keep-alive` - це заголовок, який вказує, що клієнт бажає підтримувати активне з'єднання (keep-alive) для подальших запитів до сервера в цьому з'єднанні;

– `Upgrade-Insecure-Requests: 1` - цей заголовок вказує, що клієнт бажає оновлювати незахищені запити на захищені (наприклад, з HTTP на HTTPS);

– `User-Agent` - цей заголовок містить інформацію про браузер або клієнта, який надсилає запит;

– `Accept` - цей заголовок вказує на типи вмісту, який клієнт може приймати від сервера. У цьому випадку, клієнт приймає різні типи вмісту, включаючи HTML, XML, та інші;

– `Accept-Encoding` - цей заголовок вказує на методи стиснення даних, які клієнт може приймати від сервера. У цьому випадку, клієнт приймає стиснення gzip та deflate;

– `If-None-Match` - цей заголовок вказує на "ETag" значення, яке клієнт попередньо отримав для ресурсу із сервера. Клієнт запитує оновлену версію лише у випадку, якщо це значення змінилося;

– `If-Modified-Since` - цей заголовок вказує на дату та час останнього модифікованого ресурсу, який має клієнт. Клієнт запитує оновлену версію, якщо ресурс був модифікований після цієї дати і часу.

Зазначені заголовки вказують на деталі запиту клієнта до сервера і допомагають серверу розуміти, який ресурс або сторінку клієнт запитує та які параметри обміну даними він підтримує. Цей запит є повним та коректно виконується сервером.

Після обробки запиту сервер надав коректну відповідь клієнту (Рисунок 2.11).

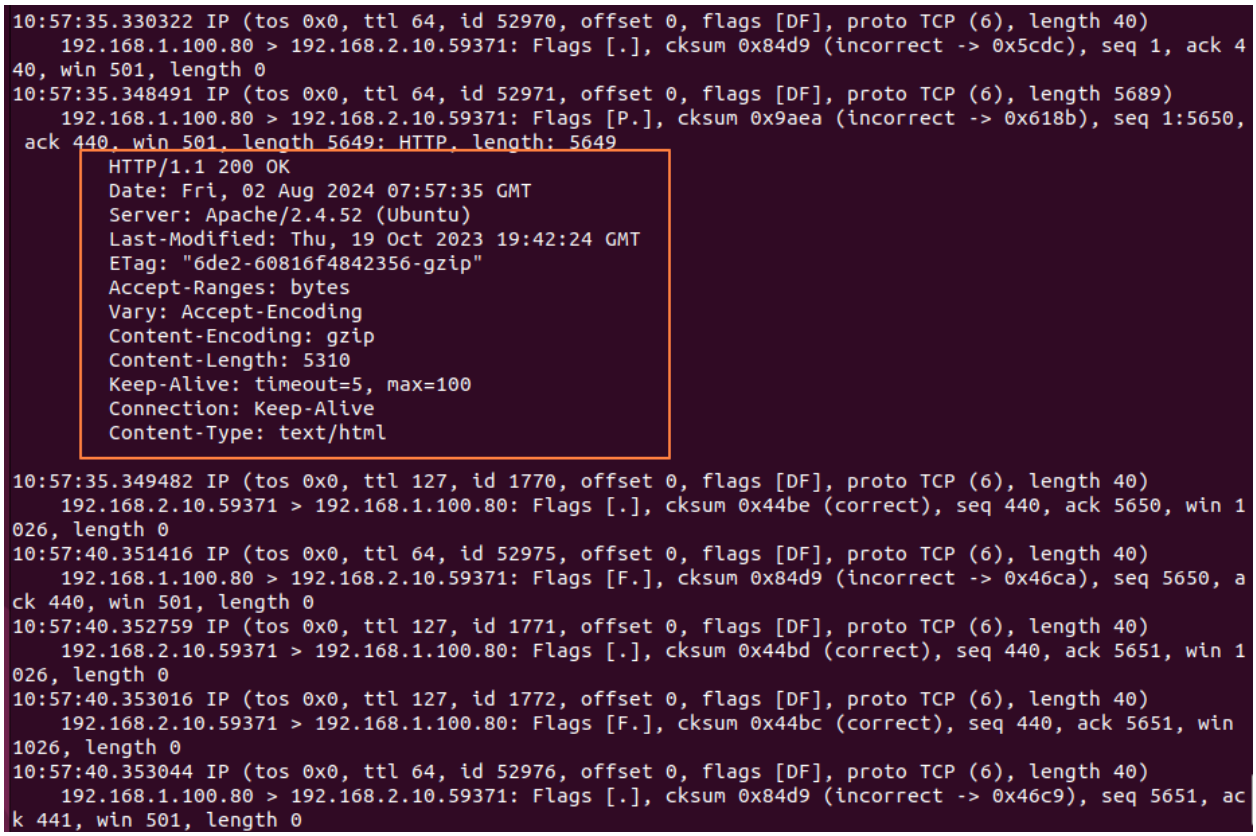

Рисунок 2.11 – Відповідь сервера на коректний запит клієнта

Значення окремих полів:

– `HTTP/1.1 200 OK` - це стрічка відповіді вказує на успішну обробку запиту. Код статусу "200 OK" означає, що запит клієнта було успішно виконано, і сервер повертає відповідь з відповідним контентом;

– `Date` - це поле вказує на дату та час генерації відповіді сервером;

– `Server` - цей заголовок ідентифікує програмне забезпечення сервера, яке обробляє запит. У цьому випадку це вебсервер Apache версії 2.4.52, що працює на операційній системі Ubuntu Linux;

– `Last-Modified` - це поле вказує на дату та час останньої модифікації ресурсу на сервері;

– `ETag` - це унікальний ідентифікатор версії ресурсу. Він використовується для оптимізації кешування. Клієнт використовує ETag для перевірки, чи змінився ресурс з моменту останнього завантаження;

– `Accept-Ranges` - це поле вказує, що сервер підтримує запити на часткове завантаження ресурсу за байтами. Це дозволяє клієнту завантажувати лише частину файлу, якщо це потрібно;

– `Keep-Alive` - це поле вказує на те, що з'єднання буде збережено відкритим (Keep-Alive) для подальших запитів протягом 5 секунд або до 100 запитів, після чого з'єднання буде закрито;

– `Connection` - це поле вказує на тип з'єднання. У вашому прикладі вказано `Keep-Alive`, що означає збереження активного з'єднання.

Ця відповідь свідчить про успішне завантаження ресурсу з сервера зі стисненням gzip та вказує на різні параметри інформації про ресурс та з'єднання.

Щоб показати методику виконання Slowloris атаки використаємо metasploit фреймворк в операційній систему для тестування безпеки Kali linux (рисунок 2.12).

```
msf6 auxiliary(dos/http/slowloris) > info

       Name: Slowloris Denial of Service Attack
     Module: auxiliary/dos/http/slowloris
    License: Metasploit Framework License (BSD)
       Rank: Normal
  Disclosed: 2009-06-17

Provided by:
  RSnake
  Gokberk Yaltirakli
  Daniel Teixeira
  Matthew Kienow <matthew_kienow[AT]rapid7.com>
```

Рисунок 2.12 – Metasploit фреймворк для виконання Slowloris атаки

Вивід команди `info` показує основні параметри та налаштування для атаки (рисунок 2.13).

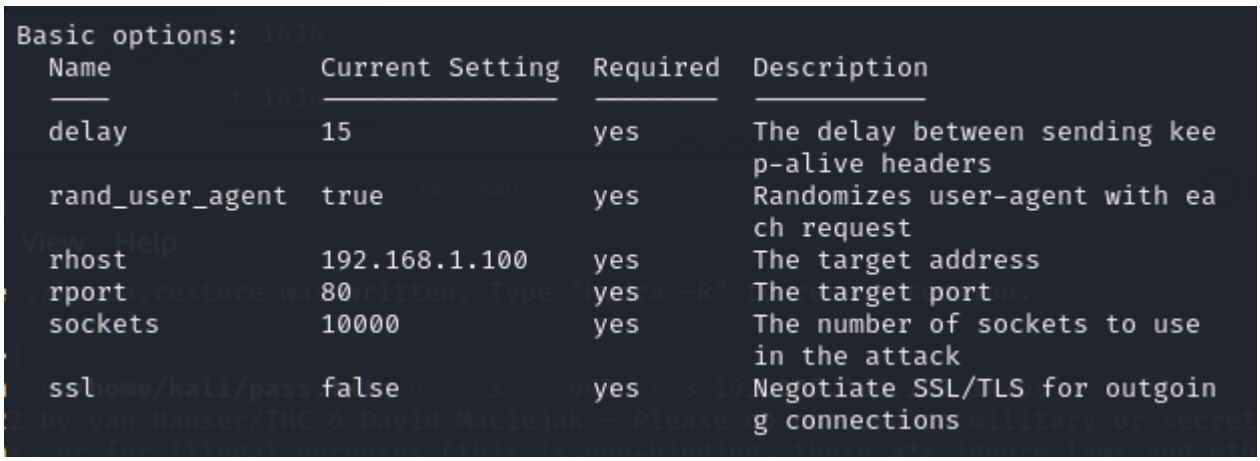

Рисунок 2.13 – Основні параметри та налаштування metasploit

Ці параметри дозволяють налаштовувати параметри атаки Slowloris, визначаючи часові і мережеві параметри, такі як затримка між запитами, кількість сокетів та інші

Докладний опис основних параметрів модуля auxiliary/dos/http/slowloris у Metasploit:

– `delay` - цей параметр визначає затримку (час у секундах), яка встановлюється між відправкою заголовків keep-alive. В атаках Slowloris затримка між заголовками допомагає тримати з'єднання відкритими і залишати їх незавершеними;

– `rand_user_agent` - цей параметр вказує, чи потрібно випадковим чином змінювати User-Agent з кожним запитом. User-Agent - це інформація, яку веб-клієнт надсилає на вебсервер для ідентифікації себе. Використання випадкового User-Agent може ускладнити виявлення атаки;

– `rhost` - цей параметр вказує цільову IP-адресу вебсервера, на який буде направлена атака Slowloris;

– `rport` - цей параметр визначає цільовий порт вебсервера, на який буде виконуватися атака;

– `sockets` - цей параметр вказує кількість сокетів (мережевих з'єднань), які будуть використовуватися під час атаки. Збільшення кількості

соксетів може збільшити ефективність атаки, але також може вимагати більше ресурсів;

– `ssl (SSL/TLS)` - цей параметр вказує, чи слід використовувати SSL/TLS для вихідних з'єднань.

Для збільшення інтенсивності атаки було використано дві операційні Kali linux з однаковими параметрами налаштування атаки. Виконаємо запуск атаки за допомогою команди `msf6 auxiliary(dos/http/slowloris) > exploit.`

Тепер давайте докладніше розглянемо взаємодію клієнта з сервером при здійсненні Slowloris атаки.

У відповідь на запити від клієнтів вебсервер відкриває тимчасові з'єднання або "слоти". У випадку Slowloris атаки, зловмисники намагаються зайняти всі доступні слоти, штучно створюючи багато з'єднань до сервера. Вони не завершують з'єднання, але також не надсилають повних запитів, тобто не завершують HTTP-запит.

З виводу команди `tcpdunp` на вебсервері можна побачити що клієнт надіслав незавершений запит до сервера (рисунок 2.14)

```
    192.168.0.105.45636 > 192.168.1.100.80: Flags [S], cksum 0x737d (correct), seq 2861801176, win 642
40, options [mss 1460,sackOK,TS val 3599664100 ecr 0,nop,wscale 7], length 0
12:47:56.475305 IP (tos 0x0, ttl 63, id 45039, offset 0, flags [DF], proto TCP (6), length 172)
    192.168.0.105.45632 > 192.168.1.100.80: Flags [P.], cksum 0xbdd0 (correct), seq 21:141, ack 1, win
 502, options [nop,nop,TS val 3599664100 ecr 1829395491], length 120: HTTP
12:47:56.475348 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 60)
    192.168.1.100.80 > 192.168.0.105.45636: Flags [S.], cksum 0x834c (incorrect -> 0xabca), seq 122950
0818, ack 2861801177, win 65160, options [mss 1460,sackOK,TS val 1829395492 ecr 3599664100,nop,wscale
7], length 0
12:47:56.475393 IP (tos 0x0, ttl 64, id 53148, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.1.100.80 > 192.168.0.105.45632: Flags [.], cksum 0x8344 (incorrect -> 0x303e), seq 1, ack
141, win 509, options [nop,nop,TS val 1829395492 ecr 3599664100], length 0
12:47:56.475882 IP (tos 0x0, ttl 63, id 25864, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.0.105.45636 > 192.168.1.100.80: Flags [.], cksum 0xd728 (correct), seq 1, ack 1, win 502,
options [nop,nop,TS val 3599664101 ecr 1829395492], length 0
12:47:56.475926 IP (tos 0x0, ttl 63, id 25865, offset 0, flags [DF], proto TCP (6), length 72)
    192.168.0.105.45636 > 192.168.1.100.80: Flags [P.], cksum 0x9e00 (correct), seq 1:21, ack 1, win 5
02, options [nop,nop,TS val 3599664101 ecr 1829395492], length 20: HTTP, length: 20
        GET /?166 HTTP/1.1
12:47:56.475975 IP (tos 0x0, ttl 64, id 37059, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.1.100.80 > 192.168.0.105.45636: Flags [.], cksum 0x8344 (incorrect -> 0xd70d), seq 1, ack
21, win 509, options [nop,nop,TS val 1829395492 ecr 3599664101], length 0
12:47:56.476344 IP (tos 0x0, ttl 63, id 26269, offset 0, flags [DF], proto TCP (6), length 60)
    192.168.0.105.45650 > 192.168.1.100.80: Flags [S], cksum 0x0e0a (correct), seq 4045913512, win 642
40, options [mss 1460,sackOK,TS val 3599664101 ecr 0,nop,wscale 7], length 0
12:47:56.476386 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 60)
    192.168.1.100.80 > 192.168.0.105.45650: Flags [S.], cksum 0x834c (incorrect -> 0xb398), seq 155878
478, ack 4045913513, win 65160, options [mss 1460,sackOK,TS val 1829395493 ecr 3599664101,nop,wscale 7
], length 0
12:47:56.477209 IP (tos 0x0, ttl 63, id 25866, offset 0, flags [DF], proto TCP (6), length 214)
    192.168.0.105.45636 > 192.168.1.100.80: Flags [P.], cksum 0x20f7 (correct), seq 21:183, ack 1, win
 502, options [nop,nop,TS val 3599664102 ecr 1829395492], length 162: HTTP
```

Рисунок 2.14 – Приклад незавершеного запиту до сервера при виконанні атаки

Сервер, очікуючи завершення запиту, тримає відкриті з'єднання для атакуючих клієнтів.

За певний період часу незавершені з’єднання сервер скидає зі статусом "408 Request Timeout". (рисунок 2.15)

Рисунок 2.15 – Скидання з’єднання сервером зі статусом "408 Request Timeout" при виконанні атаки

Відповідь сервера зі статусом "408 Request Timeout говорить про те, що сервер отримав запит від клієнта, але не отримав очікуваного продовження HTTP-запиту від клієнта вчасно.

З виводу команди `top` (рисунок 2.16) можна побачити що під час атаки в загальному сервер працює в нормальному режимі. Відсоток простою процесора складає 99,2 % і також достатньо вільної пам’яті.

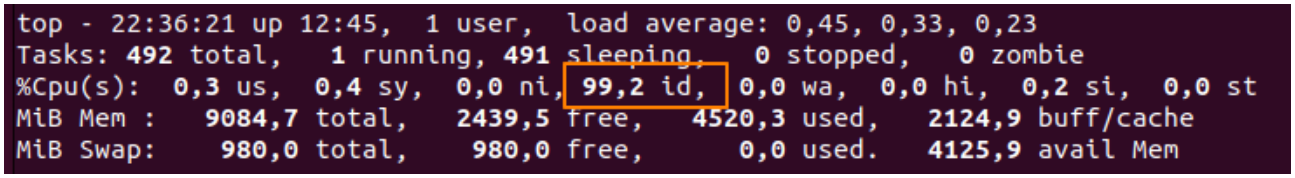

Рисунок 2.16 – Вивід команди `top`

З наданої інформації командою `top` можна зробити висновок що під час атаки ресурси сервера (процесорний час, пам'ять) не вичерпуються стрімко.

Використаємо утиліту iptraf-ng, як монітор мережевого трафіку в операційній системі Ubuntu Linux [30]. Ця утиліта надає можливість в режимі реального часу відстежувати і аналізувати мережевий трафік, який проходить через мережеві інтерфейси системи. Вона надає детальну інформацію про мережевий трафік, який проходить через мережеві інтерфейси. Можна відстежувати обсяги вхідного та вихідного трафіку, а також розподіл трафіку за протоколами.

Утиліта працює в режимі реального часу, що дозволяє спостерігати за трафіком на льоту. Можна перевіряти активність мережі в момент часу та слідкувати за змінами в режимі реального часу. iptraf-ng дозволяє переглядати, які протоколи використовуються в мережі. Можна побачити, скільки даних передається через HTTP, FTP, SSH, ICMP і багато інших протоколів.

Можна вибирати, які мережеві інтерфейси моніторити. iptraf-ng надає загальну статистику щодо кожного мережевого інтерфейсу, включаючи обсяги вхідного та вихідного трафіку, кількість пакетів, помилок.

Можна сортувати мережевий трафік за джерелом та призначенням, що дозволяє визначити, які пристрої взаємодіють в мережі.

На рисунку 2.17 можна побачити що за проміжок часу 60 секунд (Time: 0:01) є велика кількість вхідних з’єднань TCP порт 80, але кількість пакетів та обсяг трафіку є мізерним.

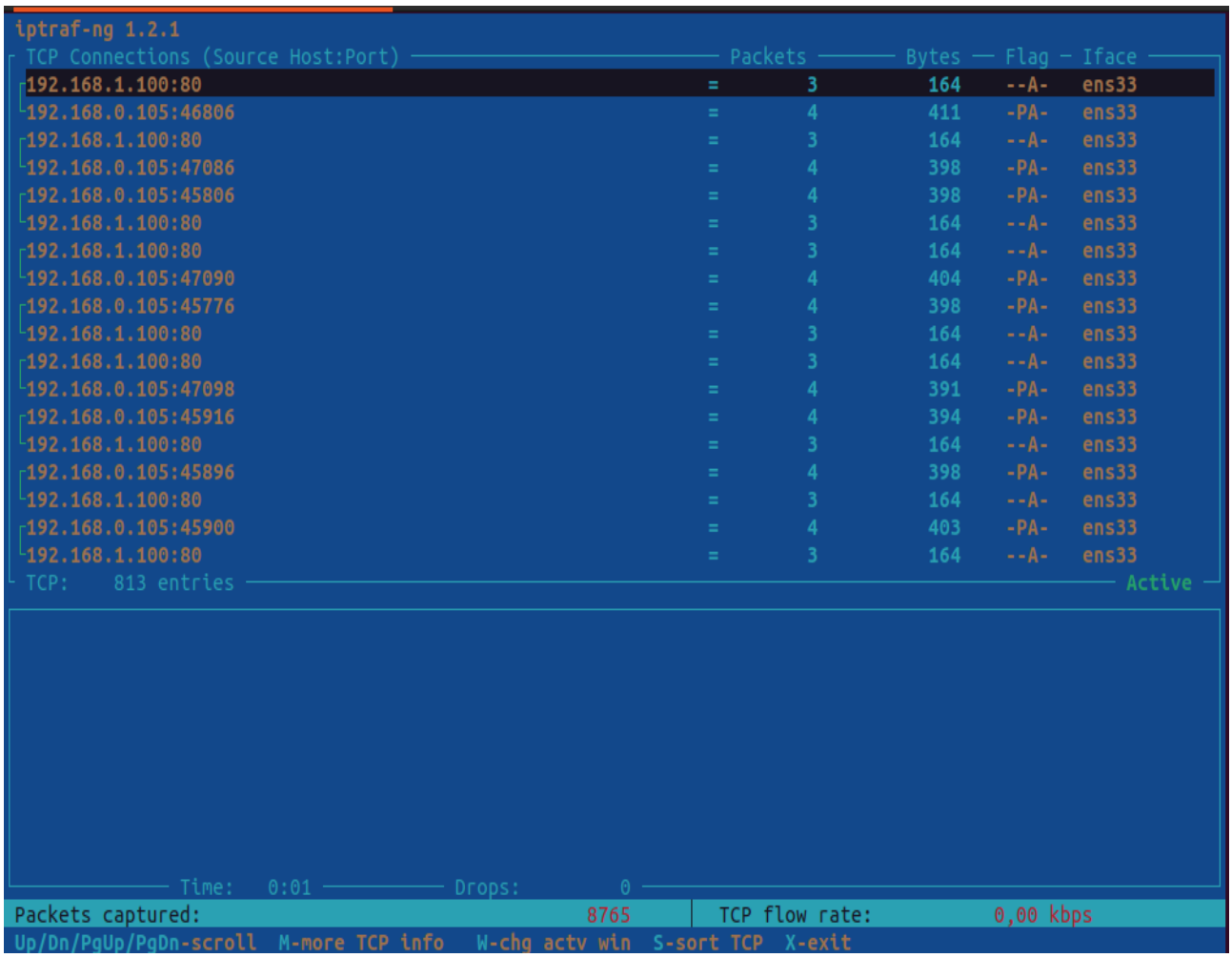

Рисунок 2.17 – Вивід статистики утилітою `iptraf-ng`

Велика кількість відкритих сесій на порт 80 TCP з мізерним обсягом трафіку та не великою кількістю пакеті в кожній сесії якраз і є признакам атаки Slowloris. При такій інтенсивності атаки вебсервер неспроможний обробляти запити від реальних користувачів і відмовляє в обслуговуванні. Це можна спостерігати, спробувавши завантажити тестову сторінку з операційної системи Windows 11, яка розташовану в локальній мережі (рисунок 2.18).

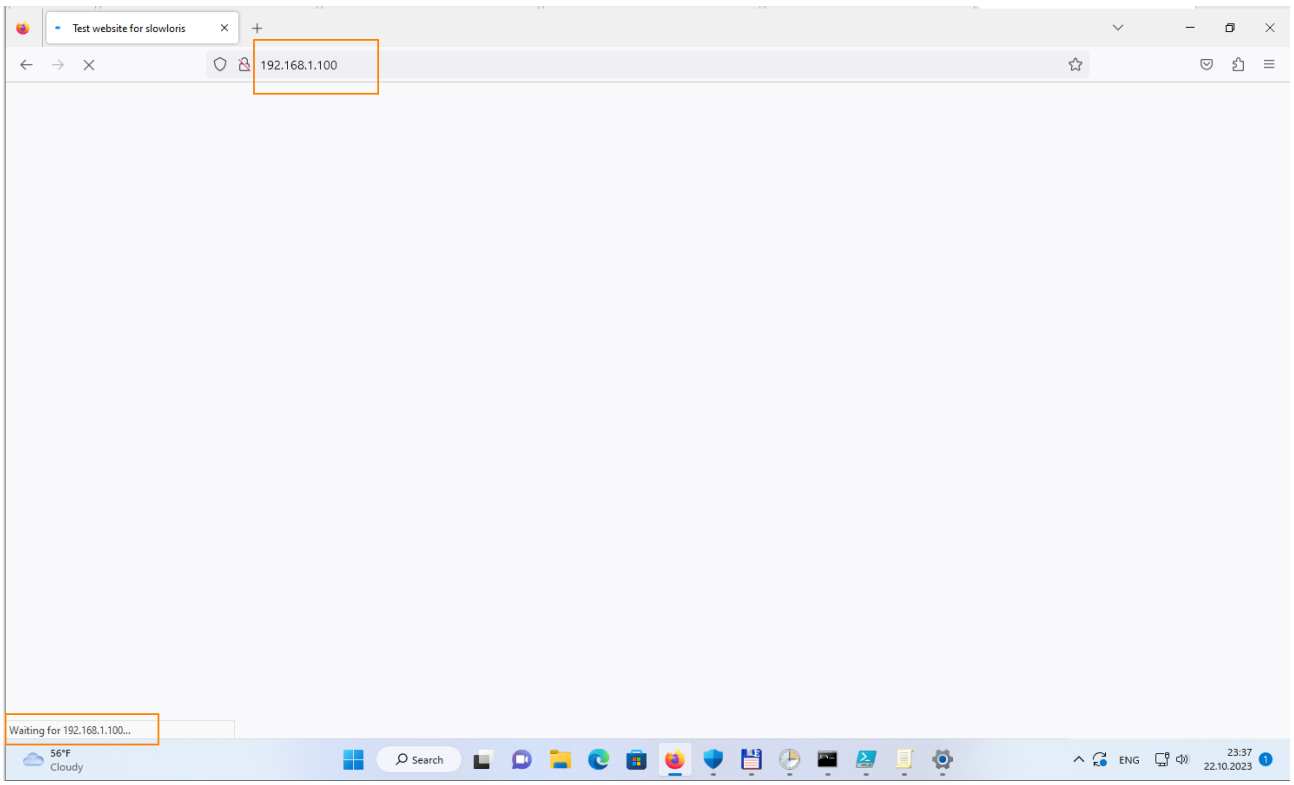

Рисунок 2.18 – Наслідки атаки Slowloris на вебсервер

Отже атака пройшла успішно. Вебсервер перевантажений запитами та не відповідає на запити користувачів локальної мережі.

В додатку Б наведено вміст лог файлу access.log сервера Apache. В якому можна також відслідкувати атаку Slowloris з Kali Linux (IP 192.168.0.105) та коректне завантаження сторінки з Windows 11 (IP 192.168.2.10).

3 РОЗРОБКА ТА ОЦІНКА ФУНКЦІОНАЛЬНОСТІ ЗАХИСТУ

3.1 Алгоритм та програмне забезпечення захисту

Розробка алгоритму виявлення атаки Slowloris та програмного забезпечення для захисту вимагає комплексного підходу та інтеграції з різними рівнями інфраструктури сервера. Оскільки атака Slowloris полягає у використанні багатьох незавершених HTTP-запитів для перевантаження цільового сервера то моніторинг в реальному часі кількості з'єднань з однієї IP адреси в стані ESTABLISHED на порти 80 та 443 TCP допоможе виявити атаку. Програмне забезпечення повинно виявляти IP-адреси, що ініціюють атаку, та блокувати їх шляхом використання брандмауера iptables [31]. Цей сервіс повинен постійно моніторити та аналізувати надходження трафіку, і якщо будуть виявлені ознаки Slowloris, автоматично додавати IP-адреси до списку заборонених у iptables. Реалізація можливості сповіщення про виявлення атаки в Telegram Bot дозволить оперативно інформувати адміністратора сервера про здійснення атаки [32]. Розробка функціоналу для запуску розробленого програмного захисту як сервісу дозволить автоматично вмикати його при запуску сервера та постійно моніторити трафік.

3.1.1 Алгоритм виявлення та блокування Slowloris атак

Ефективний захист вебсерверів від атак Slowloris вимагає чіткої стратегії, яка поєднує моніторинг трафіку, швидке реагування на аномалії та автоматизацію процесів блокування. Розроблений алгоритм виявлення та блокування Slowloris атак забезпечує комплексний підхід до мінімізації їхнього впливу на вебсервери.

Блок-схема алгоритму роботи показана на рисунку 3.1.

Рисунок 3.1 – Алгоритм роботи механізму блокування Slowloris атаки

Алгоритм виявлення та блокування Slowloris включає наступну послідовність:

1) Моніторинг з'єднань. Сервіс моніторить кількість встановлених з'єднань з однієї IP-адреси на портах 80 та 443 TCP у стані ESTABLISHED.

2) Аналіз аномалій. Програма аналізує трафік, шукаючи ознаки Slowloris атаки, такі як аномальне збільшення кількості з'єднань.

3) Виявлення IP-адрес, що ініціюють атаку. При виявленні характерних ознак Slowloris атаки, програма ідентифікує IP-адреси, з яких надходять запити.

4) Блокування IP-адрес в iptables. Отримавши IP, програма автоматично додає до списку заборонених у iptables для блокування всіх спроб з'єднання з цього джерела.

5) Сповіщення адміністратора. Розроблений функціонал сповіщення в Telegram Bot повідомляє адміністратора про виявлення IP-адрес, що здійснють Slowloris атаку.

6) Інтеграція як сервісу. Програмний захист працює як сервіс, що автоматично запускається під час старту сервера, надаючи постійний моніторинг трафіку та застосування заходів захисту.

7) Під час виявлення підозрілих IP-адрес, програма веде журнал подій, фіксуючи деталі про кожну заборонену адресу. Журнал подій буде корисним для аналізу, моніторингу та вдосконалення захисту, дозволяючи адміністраторам серверів відслідковувати історію, розуміти та вдосконалювати заходи захисту, а також вживати необхідні заходи для уникнення майбутніх атак.

Цей алгоритм надає автоматичний та ефективний механізм для виявлення та блокування атаки Slowloris, забезпечуючи оперативну реакцію та захист від цієї форми DOS атак на вебсервери.

3.1.2 Встановлення та налаштування брандмауера iptables

Iptables - це програмний брандмауер для операційних систем на базі ядра Linux [33]. Він використовується для керування трафіком в мережі шляхом визначення правил фільтрації пакетів. Iptables надає засоби для налаштування правил фільтрації, маскування IP-адрес (NAT), трансляції портів та багато інших можливостей для контролю мережевого трафіку.

Брандмауер iptables має можна встановлювати правила для приймання, відхилення або перенаправлення пакетів на основі різних параметрів, таких як IP-адреси джерела та призначення, порти, протоколи тощо. Iptables дозволяє використовувати NAT для перетворення IP-адресів в мережі, що дозволяє приховувати внутрішню структуру мережі від зовнішнього світу. Є можливість налаштовувати трансляцію портів для перенаправлення пакетів на інші порти або сервери в мережі.

Iptables - потужний інструмент для забезпечення безпеки та керування трафіком в операційній системі Linux і є важливою частиною системи безпеки для багатьох серверів та маршрутизаторів. Тому для можливості виконання алгоритму було використано брандмауер iptables.

Для можливостівикористання брандмауера iptables на операційній системі Ubuntu потрібно виконати наступні кроки:

1) Встановити брандмауер iptables на Ubuntu за допомогою пакетного менеджера apt.

```
#apt install iptables
```

2) Встановити пакет iptables-persistent на Ubuntu.

```
#apt install iptables-persistent
```

Команда встановлює пакет iptables-persistent в системі Ubuntu. Цей пакет дозволяє зберігати правила брандмауера iptables та автоматично завантажувати їх при перезапуску системи. Після встановлення iptables-persistent є можливість зберегти поточні налаштування брандмауера або додати правила, які будуть завантажені при наступному запуску системи.

3) Запустити службу iptables за допомогою systemd та додати її в автозавантаження.

```
#systemctl start iptables
#systemctl enable iptables
```

4) Здійснити перевірку статусу служби iptables за допомогою команди `systemctl status iptables` . На рисунку 3.2 показано вивід цієї команди в Ubuntu.

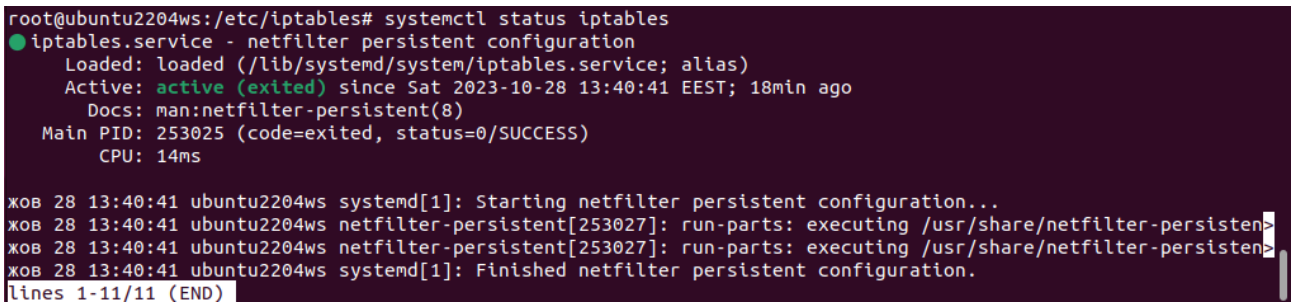

Рисунок 3.2 – Вивід команди `systemctl status iptables`

Щоб переглянути поточні налаштування можна скористатись командою `iptables -nvL --line-numbers`.

Ця команда використовується для відображення поточних правил брандмауера iptables у вигляді списку разом з порядковими номерами правил.

Давайте розглянемо, що означають окремі частини цієї команди:

1) `iptables` – це виклик інструменту управління брандмауером iptables;

2) `-nvL` - це параметри, де -n вимагає показувати порти і IP-адреси в числовому форматі, а -v показує детальну інформацію про правила. Опція -L вказує вивести список правил брандмауера.

3) `--line-numbers` вимагає відображення порядкових номерів правил.

На рисунку 3.3 показано приклад виводу команди `iptables -nvL --line-numbers`.

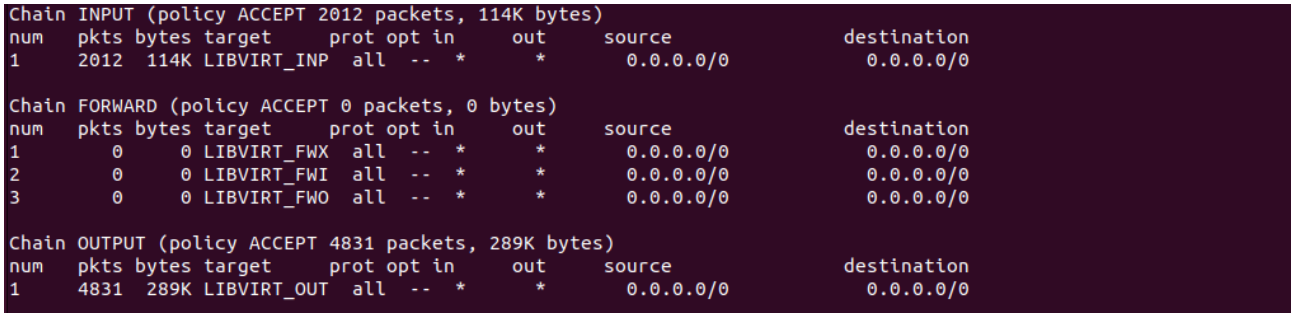

Рисунок 3.3 – Вивід команди `iptables -nvL --line-numbers`

Встановлення та налаштування брандмауера iptables на Ubuntu дозволить контролювати та керувати трафіком, включаючи захист від певних типів атак, в тому числі Slowloris, шляхом блокування підозрілого трафіку або заздалегідь визначених портів та IP-адрес.

3.1.3 Вибір програмного середовища розробки

Вибір програмного середовища розробки для написання програмного забезпечення захисту від Slowloris атак є одним з ключових моментів побудови та впровадження системи захисту та сповіщення від даного типу атак.

У даному проекті використано мову програмування C.

Мова програмування C широко використовується для розробки різноманітних програм, включаючи операційні системи, драйвери, вбудовані системи, програмне забезпечення високої продуктивності та багато іншого [16].

Мова C відома своєю швидкодією та ефективністю. Вона надає прямий доступ до пам'яті та відповідність мови машини, що дозволяє оптимізувати програми для швидкої роботи.

Код, написаний на C, зазвичай легко переноситься між різними платформами, оскільки вона забезпечує низький рівень абстракції, що дозволяє ефективно працювати на різних архітектурах. C надає прямий доступ до пам'яті та дозволяє розробляти драйвери, операційні системи та вбудоване програмне забезпечення.

У Ubuntu за замовчуванням встановлений компілятор GCC, який дозволяє компілювати програми написані на C за допомогою командного рядка [34]. Його можна викликати командою `gcc`. Є багато IDE, які підтримують мову C в Ubuntu, наприклад, Code::Blocks, CLion, NetBeans, або Eclipse. Також є текстові редактори з підтримкою C такі як vim, emacs, gedit або mcedit з розширеннями для мови C.

В даному проекті буде використовуйте текстовий редактор mcedit для створення файлу з розширенням .c та написання коду програми.

3.1.4 Написання програмного забезпечення та його компіляція

Програмне забезпечення написане на мові C та призначене для моніторингу, виявлення, блокування та інформування про Slowloris атаку на вебсервер. Лістинг вихідного файлу програми приведено в додатку В.

На рисунку 3.4 показано фрагмент коду програми включення необхідних заголовкових файлів і визначення макросу.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>

#define MAX_LINE_LENGTH 256
```

Рисунок 3.4 – Початок коду програми

Код починається з підключення чотирьох стандартних заголовкових файлів. Файл stdio.h надає функції для роботи з введенням та виведенням, такими як printf і fopen. Файл string.h додає функції для роботи з рядками, такі як strstr і strtok. Файл stdlib.h містить стандартні бібліотечні функції для управління пам'яттю, процесами та іншими загальними завданнями, наприклад, malloc і system. Файл time.h включає функції для роботи з часом і датою, такі як time, strftime, і localtime. Також у коді визначено макрос MAX_LINE_LENGTH, який встановлює максимальну довжину рядка на 256 символів. Цей макрос використовується для створення буферів, що дозволяє обмежити розмір рядків, які будуть оброблятися програмою, запобігаючи переповненню буфера.

На рисунку 3.5 показано фрагмент коду програми, який призначений для читання конфігураційних значень з файлу та збереження їх у відповідні змінні.

```c
int readConfig(const char *filename, int *BLOCK_THRESHOLD, int *MAX_IP_COUNT) {
    FILE *config_file = fopen(filename, "r");
    if (config_file == NULL) {
        perror("Failed to open configuration file");
        return 1;
    }

    char line[MAX_LINE_LENGTH];

    while (fgets(line, sizeof(line), config_file) != NULL) {
        if (strstr(line, "BLOCK_THRESHOLD=") != NULL) {
            sscanf(line, "BLOCK_THRESHOLD=%d", BLOCK_THRESHOLD);
        } else if (strstr(line, "MAX_IP_COUNT=") != NULL) {
            sscanf(line, "MAX_IP_COUNT=%d", MAX_IP_COUNT);
        }
    }

    fclose(config_file);
    return 0;
}
```

Рисунок 3.5 – Фрагмент коду функції readConfig

Функція приймає назву файлу конфігурації як аргумент, а також вказівники на змінні, куди буде збережено значення параметрів BLOCK_THRESHOLD та MAX_IP_COUNT. Функція намагається відкрити файл конфігурації у режимі читання. Якщо файл не вдається відкрити, виводиться повідомлення про помилку, і функція повертає значення 1, що вказує на помилку. Якщо файл відкрито успішно, функція починає читати його вміст рядок за рядком за допомогою функції fgets. Для кожного рядка функція перевіряє, чи містить він параметр BLOCK_THRESHOLD або MAX_IP_COUNT за допомогою функції strstr. Якщо такий параметр знайдено, використовується sscanf, щоб вилучити числове значення і зберегти його у відповідну змінну (BLOCK_THRESHOLD або MAX_IP_COUNT). Після того як всі рядки файлу будуть оброблені, функція закриває файл за допомогою fclose. Якщо конфігураційний файл був успішно прочитаний, функція повертає 0, що вказує на успішне виконання.

Ця функція дозволяє динамічно налаштовувати параметри програми, зчитуючи їх із зовнішнього файлу конфігурації.

На рисунку 3.6 показано вміст конфігураційного файлу `/etc/doswww.conf`.

```
root@ubuntu2204ws:/home/admin/doswww# cat /etc/doswww.conf
MAX_IP_COUNT=100
BLOCK_THRESHOLD=100
```

Рисунок 3.6 – Вміст конфігураційного файлу `/etc/doswww.conf`

Конфігураційний файл doswww.conf містить обмеження для кількості IP-адрес MAX_IP_COUNT та порогу спрацювання BLOCK_THRESHOLD, що використовуються для виявлення атаки Slowloris.

На рисунку 3.7 показано фрагмент коду програми, який є початком функції main().

```c
int main() {
    int BLOCK_THRESHOLD = 0;
    int MAX_IP_COUNT = 0;

    if (readConfig("/etc/doswww.conf", &BLOCK_THRESHOLD, &MAX_IP_COUNT) != 0) {
        return 1;
    }

    FILE *file = popen("netstat -an", "r");
    if (file == NULL) {
        perror("Failed to run netstat");
        return 1;
    }

    char line[MAX_LINE_LENGTH];

    // Arrays to store unique IP addresses and their counts
    char *unique_ips[MAX_IP_COUNT];
    int ip_counts[MAX_IP_COUNT];
    int ip_blocked[MAX_IP_COUNT];
    for (int i = 0; i < MAX_IP_COUNT; i++) {
        unique_ips[i] = NULL; // Initialize array with NULL pointers
        ip_counts[i] = 0;
        ip_blocked[i] = 0;  // Initialize blocked flag to 0
    }
```

Рисунок 3.7 – Фрагмент коду читання конфігураційного файлу, оголошення змінних та ініціалізації масивів

У даному фрагменті коду визначаються дві змінні BLOCK_THRESHOLD і MAX_IP_COUNT, які ініціалізуються значенням 0. Ці змінні використовуються для зберігання порогу блокування IP-адрес і максимальної кількості IP-адрес, які можуть бути оброблені. Далі функція readConfig() зчитує конфігураційні значення з файлу /etc/doswww.conf. Наступний блок відкриває команду netstat -an для читання через канал за

допомогою popen(). Потім створюється буфер line розміром MAX_LINE_LENGTH для зчитування рядків із вихідного потоку команди netstat. Після цього ініціалізуються три масиви: unique_ips для зберігання унікальних IP-адрес, ip_counts для підрахунку з'єднань для кожної IP-адреси, і ip_blocked для відстеження заблокованих IP-адрес. Усі елементи масивів ініціалізуються значеннями за замовчуванням. Це готує масиви для подальшого оброблення інформації про IP-адреси, отримані з команди netstat.

На рисунку 3.8 показано фрагмент коду програми, який виконує аналіз мережевого трафіку.

```
    while (fgets(line, sizeof(line), file) != NULL) {
        // Check if the line contains the desired ports and "ESTABLISHED"
        if ((strstr(line, ":443") != NULL || strstr(line, ":80") != NULL) && strstr(line,
"ESTABLISHED") != NULL) {
            char *token = strtok(line, " ");
            int i = 0;
            while (token != NULL) {
                if (i == 4) { // Check for the fifth column (remote address)
                    // Assuming the IP address is in the format IP:PORT
                    char *ip_port = strtok(token, ":");
                    int existing_ip_index = -1;
                    for (int j = 0; j < MAX_IP_COUNT; j++) {
                        if (unique_ips[j] != NULL && strcmp(unique_ips[j], ip_port) == 0) {
                            existing_ip_index = j;
                            break;
                        }
                    }
                    if (existing_ip_index == -1) {
                        // New unique IP address
                        for (int j = 0; j < MAX_IP_COUNT; j++) {
                            if (unique_ips[j] == NULL) {
                                unique_ips[j] = strdup(ip_port);
                                ip_counts[j] = 1;
                                ip_blocked[j] = 0;  // Initialize blocked flag to 0
                                break;
                            }
                        }
                    } else if (!ip_blocked[existing_ip_index]) {  // Check if IP hasn't been
blocked
```

Рисунок 3.8 – Фрагмент коду аналізу мережевого трафіку

За допомогою даного коду виявляються з'єднання на портах 80 і 443, які знаходяться в стані "ESTABLISHED". Він зчитує кожний рядок з виводу команди netstat -an, перевіряючи, чи містить він потрібні порти та статус з'єднання.

Якщо рядок відповідає цим критеріям, він розбивається на токени (частини рядка, розділені пробілами) за допомогою функції strtok(). Потім код перевіряє п'ятий токен, який представляє віддалену IP-адресу та порт.

Код намагається знайти IP-адресу в масиві unique_ips. Якщо IP-адреса є новою, вона додається до масиву, а лічильник підключень (ip_counts) для неї встановлюється на 1. Якщо IP-адреса вже присутня в масиві і ще не заблокована, код збільшує її лічильник підключень, перевіряючи при цьому, чи не перевищує цей лічильник поріг блокування (BLOCK_THRESHOLD). Якщо IP-адреса перевищує поріг блокування, вона підлягає подальшому обробленню для блокування.

На рисунку 3.9 показано фрагмент коду програми, який блокує IP-адресу, яка перевищила порогове значення з'єднань, додає правило до iptables, реєструє подію в лог-файлі з поточною датою та часом, позначає IP-адресу як заблоковану, і відправляє сповіщення в Telegram.

```
                    if (ip_counts[existing_ip_index] > BLOCK_THRESHOLD) {
                        // Block the IP address and log the message
                        char command[MAX_LINE_LENGTH];
                        snprintf(command, sizeof(command), "sudo iptables -A INPUT -s %s -j
DROP", unique_ips[existing_ip_index]);
                        system(command);
                        printf("Blocked IP: %s\n", unique_ips[existing_ip_index]);

                        // Log the blocked IP address along with the current date and time
                        char log_message[MAX_LINE_LENGTH];
                        time_t now;
                        struct tm *tm_info;
                        time(&now);
                        tm_info = localtime(&now);
                        strftime(log_message, MAX_LINE_LENGTH, "%Y-%m-%d %H:%M:%S",
tm_info);
                        snprintf(log_message + strlen(log_message), MAX_LINE_LENGTH -
strlen(log_message), " DOS Attack Alert: IP Blocked %s", unique_ips[existing_ip_index]);
                        logMessage("/var/log/doswww.log", log_message);

                        // Mark IP as blocked
                        ip_blocked[existing_ip_index] = 1;

                        // Send a notification to Telegram
                        sendNotification(unique_ips[existing_ip_index]);
                    }
```

Рисунок 3.9 – Фрагмент коду обробки IP-адрес зловмисників та сповіщень

В даному коді, якщо лічильник підключень для IP-адреси перевищує встановлений поріг блокування (BLOCK_THRESHOLD), код блокує цю IP-

адресу, додаючи правило до iptables. Це правило забороняє вхідні з'єднання з цієї IP-адреси за допомогою команди iptables -A INPUT -s <IP> -j DROP, яка виконується системним викликом system(). Далі код формує повідомлення для журналу, яке містить поточну дату і час, а також текст, що сигналізує про блокування IP-адреси через атаку Slowloris. Це повідомлення записується у файл журналу /var/log/doswww.log за допомогою функції logMessage(). Після цього код позначає IP-адресу як заблоковану, щоб уникнути повторних спроб її блокування.

На завершення код відправляє повідомлення з попередженням про блокування IP-адреси в Telegram, використовуючи функцію sendNotification().

На рисунку 3.10 показано фрагмент коду програми, який використовується для запису повідомлень у файл журналу та надсилання сповіщень у Telegram [35].

```
// Function for logging messages to a file
void logMessage(const char *log_filename, const char *message) {
    FILE *log_file = fopen(log_filename, "a");
    if (log_file != NULL) {
        fprintf(log_file, "%s\n", message);
        fclose(log_file);
    }

}

// Function to send a notification to Telegram
void sendNotification(const char *ip) {
    char curl_command[MAX_LINE_LENGTH];
    snprintf(curl_command, sizeof(curl_command), "curl -s -X POST https://api.telegram.org/
bot6602659373:[illegible] -d chat_id=[illegible] -d
text=\"DOS Attack Alert: IP Blocked %s\" &>/dev/null 2>&1", ip);
    system(curl_command);
}
```

Рисунок 3.10 – функції логування та надсилання сповіження в Telegram

Функція logMessage відповідає за запис повідомлень у файл журналу. Вона приймає два параметри: ім'я файлу та текст повідомлення. Спочатку функція відкриває файл у режимі додавання, що дозволяє додавати нові записи в кінець файлу без видалення існуючого вмісту. Якщо файл відкрито успішно, повідомлення записується у файл з новим рядком після нього. Після запису файл закривається, щоб завершити операцію.

Функція sendNotification використовується для відправки сповіщень в Telegram через API. Вона приймає параметр, що містить IP-адресу, яку потрібно включити в повідомлення. Функція формує команду curl, яка надсилає POST-запит до Telegram API. Команда включає токен бота, ідентифікатор чату та текст повідомлення з IP-адресою. Після формування команди вона виконується за допомогою функції system, що дозволяє виконувати команду у системному середовищі. Для запобігання виводу інформації про виконання команди на екран, всі виходи та помилки перенаправляються у /dev/null.

Ця програма дозволяє виявляти, реагувати та інформувати про атаки типу Slowloris, блокуючи IP-адреси, які генерують підозрілу активність та перевищують встановлені порогові значення.

Щоб скомпілювати програму alertdoswww.c в Ubuntu необхідно виконати наступну команду:

```
#gcc alertdoswww.c -o alertdoswww
```

Ця команда скомпілює програму `alertdoswww.c` та створить виконуваний файл `alertdoswww` у тій самій директорії.

3.1.5 Інтеграція програмного забезпечення як сервісу

Здійснимо налаштування програмного захисту як сервісу. Програма буде автоматично запускатися під час старту сервера, забезпечуючи постійний моніторинг трафіку та застосування заходів захисту.

Для інтеграції програми alertdoswww як сервісу в Ubuntu скористаємось systemd, менеджером системних служб.

На рисунку 3.11 показано файл конфігурації служби у каталозі `/etc/systemd/system/` з назвою `alertdoswww.service`.

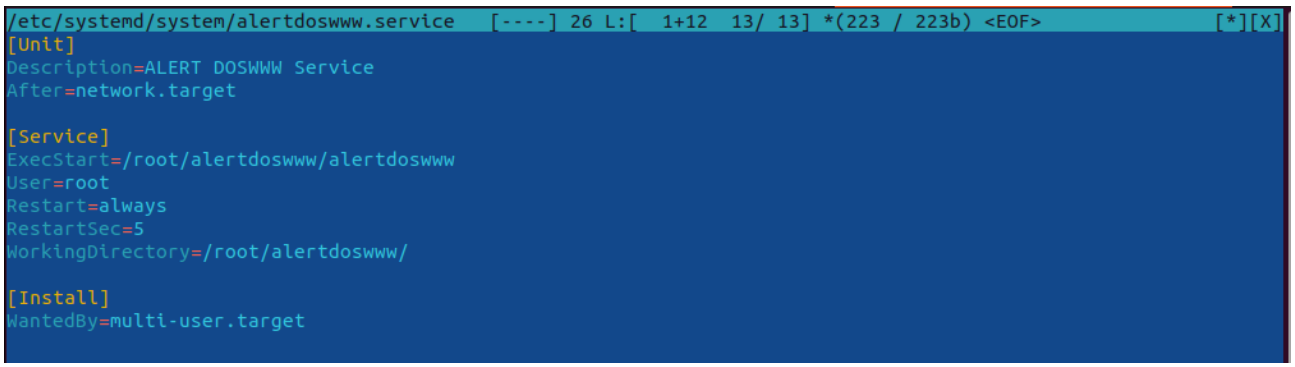

Рисунок 3.11 – Вміст файлу `alertdoswww.service`

Файл alertdoswww.service є конфігураційним файлом для systemd, який керує запуском та управлінням сервісом.

Значення розділів конфігураційного файлу наступні:

1) [Unit]. Цей розділ визначає загальні властивості служби:

- Description=ALERT DOSWWW Service – це назва служби або короткий опис її призначення;
- After=network.target - we покажчик того, що ця служба має бути запущена після досягнення стану network.target, що вказує на завершення ініціалізації мережі.

2) [Service]. Цей розділ містить налаштування для самої служб:

- ExecStart=/root/alertdoswww/alertdoswww - ця стрічка вказує на шлях до виконуваного файлу, який запускається при старті служби;
- User=root - вказує користувача, від імені якого виконується служба;
- Restart=always - ця опція вказує systemd перезапустити службу завжди, коли вона завершується;
- RestartSec=30 - ця опція вказує інтервал часу (у секундах), після якого служба буде перезапущена у випадку її завершення;
- WorkingDirectory=/root/alertdoswww/ - вказує робочий каталог, в якому виконується служба.

3) [Install]. Цей розділ містить інформацію про те, як служба повинна бути встановлена для автозапуску. Стрічка WantedBy=multi-user.target вказує,

що служба є бажаною для автоматичного запуску в багатокористувацькому режимі.

Отже, файл alertdoswww.service налаштовує службу для запуску в системі Ubuntu як сервісу, із вказанням команди, яку слід виконати, і налаштуваннями перезапуску та прав доступу.

Після створення файлу служби і його збереження потрібно запустити команди для завантаження служби і її запуску:

- `#systemctl daemon-reload;`
- `#systemctl start alertdoswww;`
- `#systemctl enable alertdoswww.`

Команда `systemctl enable` дозволить службі запускатися під час завантаження системи.

Можна перевірити статус служби за допомогою команди:

- `#systemctl status alertdoswww.`

Це покаже інформацію про статус нової служби та події журналу (рисунок 3.12).

```
root@ubuntu2204ws:/etc/systemd/system# systemctl status alertdoswww
● alertdoswww.service - ALERT DOSWWW Service
     Loaded: loaded (/etc/systemd/system/alertdoswww.service; disabled; vendor preset: enabled)
     Active: activating (auto-restart) since Sat 2023-10-28 16:33:01 EEST; 635ms ago
    Process: 278257 ExecStart=/root/alertdoswww/alertdoswww (code=exited, status=0/SUCCESS)
   Main PID: 278257 (code=exited, status=0/SUCCESS)
        CPU: 11ms
root@ubuntu2204ws:/etc/systemd/system#
```

Рисунок 3.12 – Вивід статус сервісу `alertdoswww`

Таким чином, alertdoswww буде автоматично запускатися як служба при завантаженні системи. Це дозволить легко керувати ним, забезпечуючи зручний та надійний функціонал.

3.2 Оцінка функціональності засобу захисту

На початковому етапі оцінки функціональності відключимо сервіс автоматичного відслідковування атаки за допомогою команди `systemctl`

`stop alertdoswww` та здійснимо атаку Slowloris за схемою мережі згідно пункту 2.1 та принципом виконання згідно пункту 2.6.

На рисунку 3.13 можна побачити що після старту атаки тестова сторінка перестала завантажуватись.

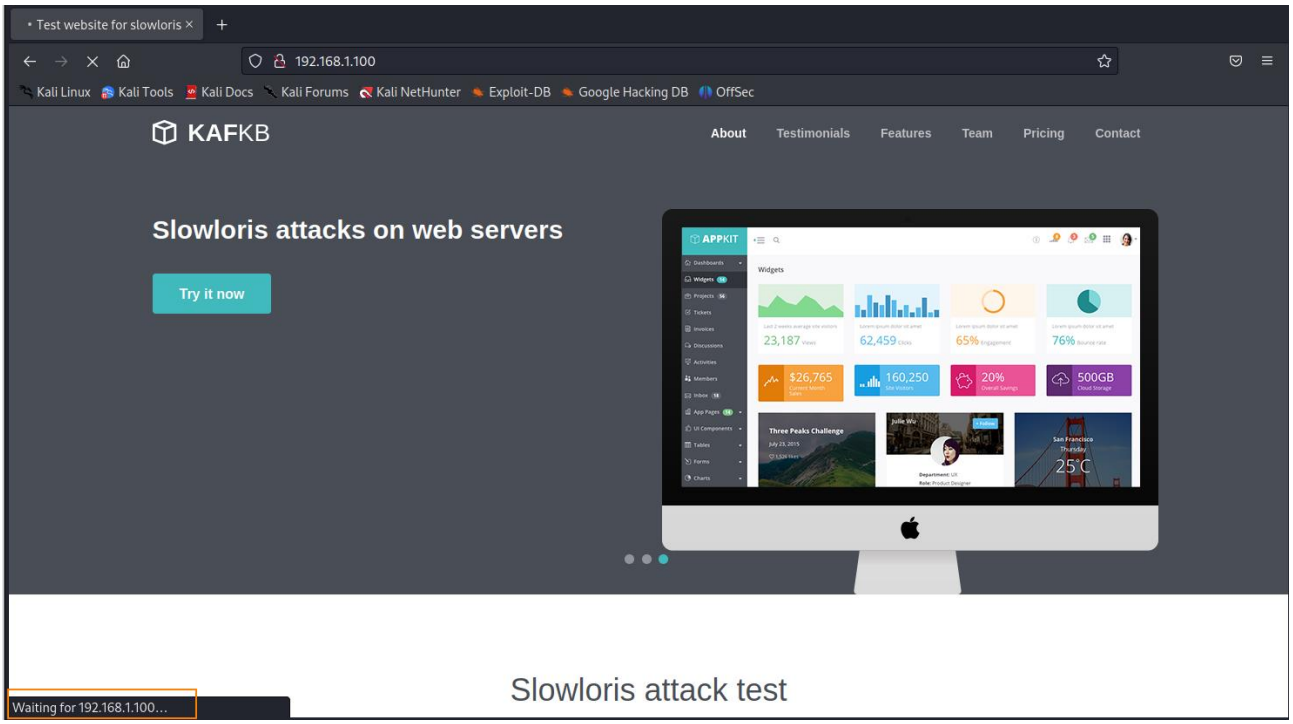

Рисунок 3.13 – Завантаження тестового сайту з операційної Kali Linux

Після ввімкнення сервісу командою `systemctl start alertdoswww` за декілька секунд ми отримаємо повідомлення в Telegram про атаку (Рисунок 3.14).

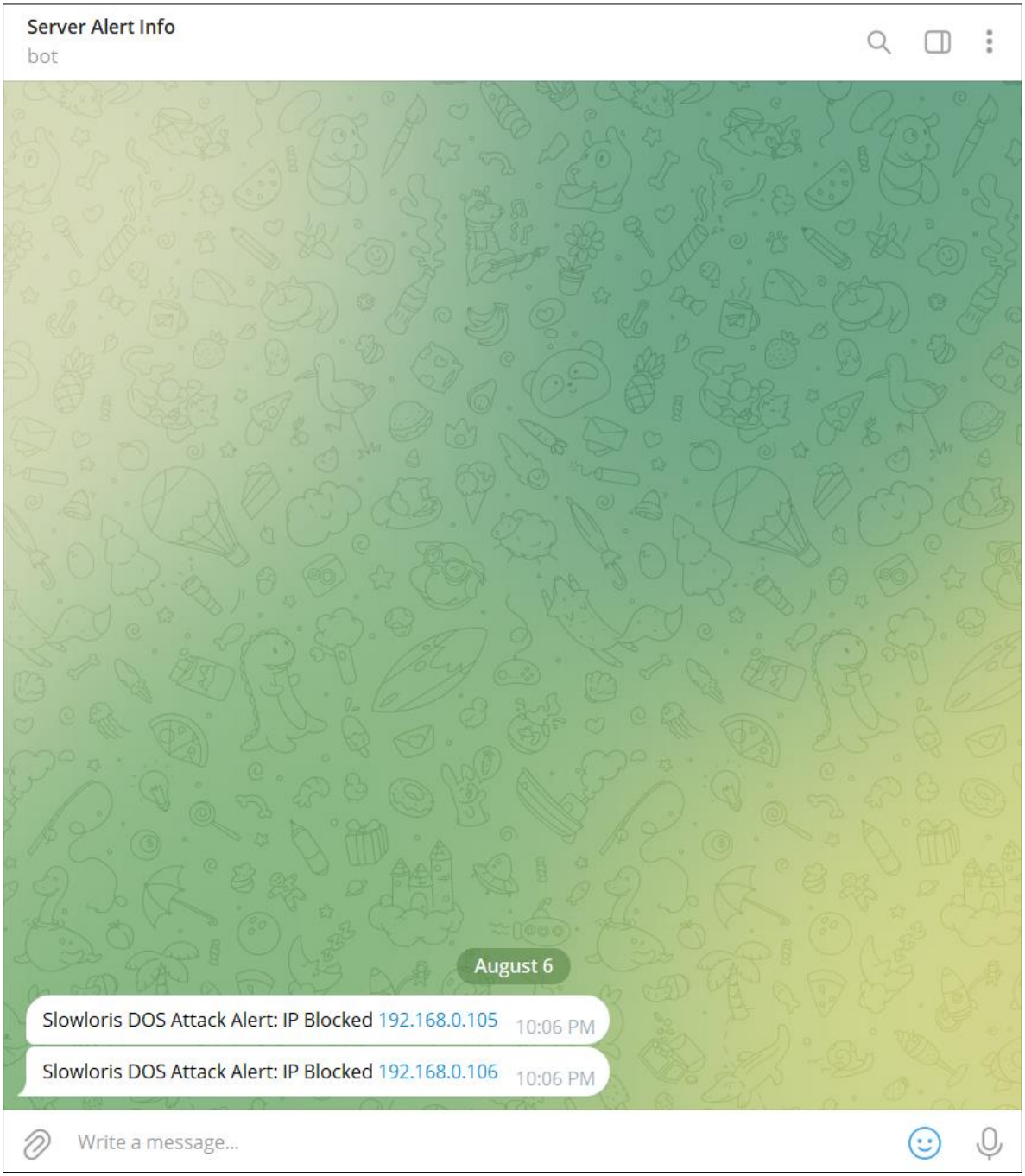

Рисунок 3.14 – Повідомлення в Telegram про атаку Slowloris

Також з виводу команди `iptables -nvL --line-numbers` (рисунок 3.15) можна побачити що IP адреси атакуючих заблоковані.

```
Chain INPUT (policy ACCEPT 125K packets, 9620K bytes)
num   pkts bytes target     prot opt in     out     source               destination
1      203K   15M LIBVIRT_INP  all  --  *      *       0.0.0.0/0            0.0.0.0/0
2     9724  631K DROP       all  --  *      *       192.168.0.105        0.0.0.0/0
3     8404  548K DROP       all  --  *      *       192.168.0.106        0.0.0.0/0
```

Рисунок 3.15 – Вивід команди `iptables -nvL --line-numbers`

У нашому випадку, правила вказують на те, що пакети, які приходять від IP-адрес 192.168.0.105 та 192.168.0.106, будуть відкинуті.

Також лог-файл `/var/log/doswww.log` містить записи щодо блокування IP-адрес під час атаки Slowloris. Кожен запис включає дату та час блокування та повідомлення про те, що IP-адреса було заблоковано через атаку Slowloris (Рисунок 3.16).

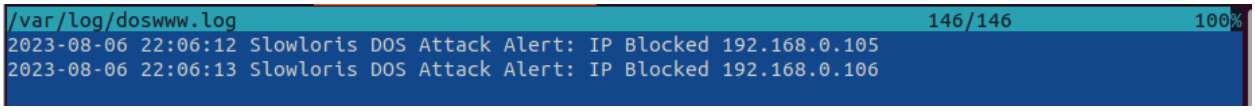

Рисунок 3.16 – Вміст лог-файлу `/var/log/doswww.log`

Як можна побачити з вмісту файлу, IP-адреси 192.168.0.105 та 192.168.0.106 були заблоковані під час атаки.

Даний лог-файл можна використовувати в системі управління інформацією про події безпеки (SIEM), щоб відстежувати атаки, виявляти загрози безпеки, та реагувати на події в мережі. SIEM системи дозволяють агрегувати, аналізувати та відображати інформацію з різних джерел журналів, включаючи такі, як doswww.log.

Також можна переконатися, що після активації системи відстеження та блокування атаки Slowloris наш вебсервер знову став доступним для користувачів. Це підтверджується завантаженням тестової сторінки (рисунку 3.17).

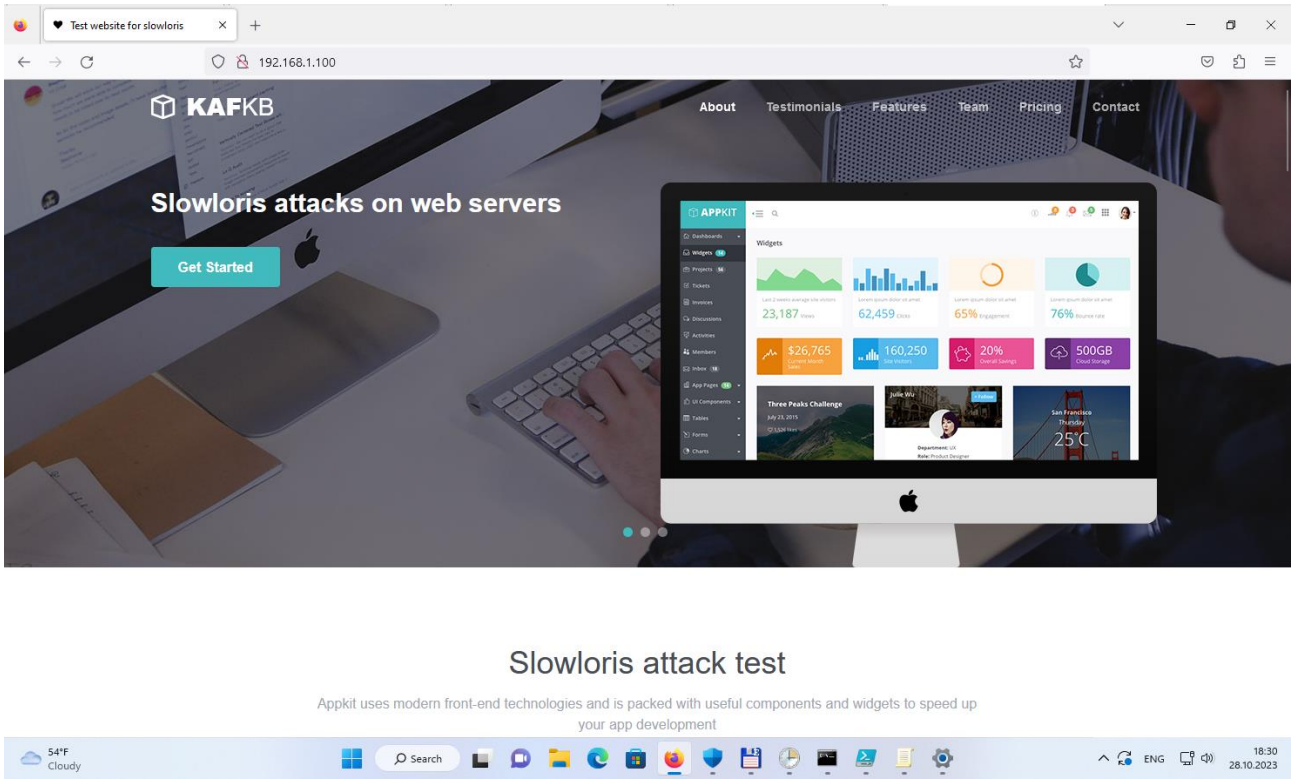

Рисунок 3.17 – Перевірка працездатності Вебсервера з операційної системи Windows 11

Це підтверджує, що система виявлення та блокування атаки Slowloris на вебсервер працює ефективно та надійно.

ВИСНОВКИ

В кваліфікаційній роботі розв’язано актуальну задачу підвищення ефективності алгоритмів виявлення Slowloris атак з розробкою та оцінкою функціональності механізму мінімізації впливу атак на вебсервери Apache. При цьому отримано наступні результати.

1. Проведено огляд методики здійснення Slowloris атак.

2. Проведено аналіз загальних методів захисту від Slowloris атак.

3. Налаштувати тестове середовище на основі гіпервізора VMware ESXi для моделювання Slowloris атаки на вебсервер.

4. Удосконалено алгоритм виявлення, блокування та сповіщення про Slowloris атаку.

5. Розроблено програмне забезпечення для виявлення Slowloris атаки та блокування IP-адрес зловмисників.

6. Реалізовано можливості запуску програмного забезпечення як сервісу для постійної роботи на вебсервері.

7. Розроблено механізм запису інформації про виявлені атаки в лог-файл для подальшого аналізу.

8. Розроблено механізм сповіщення в Telegram про здійснення атаки.

9. Проведено оцінку функціональності розробленого програмного засобу для мінімізації впливу Slowloris атак на вебсервер.

10. Підтверджено, що система виявлення та блокування атаки Slowloris на вебсервер працює ефективно та надійно.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Slowloris DDoS attack [Електронний ресурс]. — URL: https://www.cloudflare.com/learning/ddos/ddos-attack-tools/slowloris/ (дата звернення: 30.10.2024).

2. Performance comparison and analysis of slowloris, goldeneye and xerxes ddos attack tools / T. Shorey et al. 2018 international conference on advances in computing, communications and informatics (ICACCI), Bangalore, 19–22 September 2018. 2018. URL: https://doi.org/10.1109/icacci.2018.8554590

3. Generator of slow denial-of-service cyber attacks / M. Sikora et al. Sensors. 2021. Vol. 21, no. 16. P. 5473. URL: https://doi.org/10.3390/s21165473 (date of access: 21.11.2024).

4. Black S., Kim Y. An overview on detection and prevention of application layer ddos attacks. 2022 IEEE 12th annual computing and communication workshop and conference (CCWC), Las Vegas, NV, USA, 26–29 January 2022. 2022. URL: https://doi.org/10.1109/ccwc54503.2022.9720741

5. Nycz M., Szeliga T., Hajder P. Assessment of the vulnerability of the Apache server to DDoS attaks. Scientific journals of rzeszów university of technology, series: electrotechnics. 2017. P. 67–76. URL: https://doi.org/10.7862/re.2017.6.

6. SDToW: a slowloris detecting tool for wmns / V. d. S. Faria et al. Information. 2020. Vol. 11, no. 12. P. 544. URL: https://doi.org/10.3390/info11120544.

7. Mammadova K., Aslanov R. Installation of integrated intellectual information security systems in open corporate networks – DDoS attack. InterConf. 2023. No. 32(151). P. 643–651. URL: https://doi.org/10.51582/interconf.19-20.04.2023.069 (date of access: 21.11.2024).

8. What Is a Web Server? [Електронний ресурс]. — URL: https://www.hostinger.com/tutorials/what-is-a-web-server (дата звернення: 30.10.2024).

9. NGINX vs Apache – Choosing the Best Web Server in 2023 [Електронний ресурс]. — URL: https://www.hostinger.com/tutorials/nginx-vs-apache-what-to-use/ (дата звернення: 30.10.2024).

10. Developer Documentation for the Apache HTTP Server 2.4 [Електронний ресурс]. — URL: https://httpd.apache.org/docs/2.4/developer/ (дата звернення: 30.10.2024).

11. The APACHE Project / A. Sozzetti et al. EPJ web of conferences. 2013. Vol. 47. P. 03006. URL: https://doi.org/10.1051/epjconf/20134703006

12. Inside NGINX: How We Designed for Performance & Scale [Електронний ресурс]. — URL: https://www.nginx.com/blog/inside-nginx-how-we-designed-for-performance-scale/ (дата звернення: 30.10.2024).

13. Soni R. Introduction to nginx web server. Nginx. Berkeley, CA, 2016. P. 1–15. URL: https://doi.org/10.1007/978-1-4842-1656-9_1 (date of access: 21.11.2024).

14. Soni R. Nginx core architecture. Nginx. Berkeley, CA, 2016. P. 97–106. URL: https://doi.org/10.1007/978-1-4842-1656-9_5 (date of access: 21.11.2024).

15. VMware vSphere Documentation [Електронний ресурс]. — URL: https://docs.vmware.com/en/ VMware-vSphere/index.html (дата звернення: 24.10.2024).

16. Richa, Singh J. An approach to personalize vmware vsphere hypervisor (esxi) using HPE image streamer. Advances in intelligent systems and computing. Singapore, 2021. P. 363–369. URL: https://doi.org/10.1007/978-981-33-4299-6_30

17. Tymoshchuk D., Yatskiv V. Using hypervisors to create a cyber polygon. Measuring and computing devices in technological processes. 2024. No. 3. P. 52–56. URL: https://doi.org/10.31891/2219-9365-2024-79-7

18. Intel virtualization technology / R. Uhlig et al. Computer. 2005. Vol. 38, no. 5. P. 48–56. URL: https://doi.org/10.1109/mc.2005.163

19. A novel hardware assisted full virtualization technique / W. Chen et al. 2008 9th international conference for young computer scientists (ICYCS), Hunan, China, 18–21 November 2008. 2008. URL: https://doi.org/10.1109/icycs.2008.218

20. Cloud Hosted Router, CHR. URL: https://help.mikrotik.com/docs/display/ROS/Cloud+Hosted+Router%2C+CHR (дата звернення: 04.02.2024).

21. Kurniadi S. H., Utami E., Wibowo F. W. Building dynamic mesh VPN network using mikrotik router. Journal of physics: conference series. 2018. Vol. 1140. P. 012039. URL: https://doi.org/10.1088/1742-6596/1140/1/012039

22. Ubuntu Server documentation [Електронний ресурс]. — URL: https://ubuntu.com/server/docs (дата звернення: 30.10.2024).

23. Dieguez Castro J. Ubuntu. Introducing linux distros. Berkeley, CA, 2016. P. 47–72. URL: https://doi.org/10.1007/978-1-4842-1392-6_4

24. Kali Official Documentation [Електронний ресурс]. — URL: https://www.kali.org/docs/ (дата звернення: 30.10.2024).

25. Tigner M., Wimmer H., Rebman C. M. Analysis of kali linux penetration tools: a survey of hacking tools. 2021 international conference on electrical, computer and energy technologies (ICECET), Cape Town, South Africa, 9–10 December 2021. 2021. URL: https://doi.org/10.1109/icecet52533.2021.9698572

26. Metasploit Documentation [Електронний ресурс]. — URL: https://docs.metasploit.com/ (дата звернення: 30.10.2024).

27. Slowloris Denial of Service Attack - Metasploit [Електронний ресурс]. — URL: https://www.infosecmatter.com/metasploit-module-library/?mm=auxiliary/dos/http/slowloris (дата звернення: 30.10.2024).

28. Sabri S., Ismail N., Hazzim A. Slowloris dos attack based simulation. IOP conference series: materials science and engineering. 2021. Vol. 1062, no. 1. P. 012029. URL: https://doi.org/10.1088/1757-899x/1062/1/012029

29. Goyal P., Goyal A. Comparative study of two most popular packet sniffing tools-Tcpdump and Wireshark. 2017 9th international conference on computational intelligence and communication networks (CICN), Girne, 16–17 September 2017. 2017. URL: https://doi.org/10.1109/cicn.2017.8319360

30. Iptraf - Interactive Colorful IP LAN Monitor [Електронний ресурс]. — URL: https://manpages.ubuntu.com/manpages/focal/en/man8/iptraf-ng.8.html (дата звернення: 30.10.2024).

31. Basic iptables howto [Електронний ресурс]. — URL: https://help.ubuntu.com/community/IptablesHowTo (дата звернення: 30.10.2024).

32. Tymoshchuk D., Yatskiv V. Slowloris ddos detection and prevention in real-time. Theoretical and empirical scientific research: concept and trends. 2024. URL: https://doi.org/10.36074/logos-16.08.2024.036.

33. Securing Linux with a faster and scalable iptables / S. Miano et al. ACM SIGCOMM computer communication review. 2019. Vol. 49, no. 3. P. 2–17. URL: https://doi.org/10.1145/3371927.3371929

34. Gcc - GNU project C and C++ compiler [Електронний ресурс]. — URL: https://manpages.ubuntu.com/manpages/xenial/en/man1/gcc.1.html (дата звернення: 30.10.2024).

35. Ismawati D., Prasetyo I. The development of telegram BOT through short story. Brawijaya international conference on multidisciplinary sciences and technology (BICMST 2020), Malang, Indonesia, 2–3 January 2020. Paris, France, 2020. URL: https://doi.org/10.2991/assehr.k.201021.049

Міжнародний науково-технічний журнал
«Вимірювальна та обчислювальна техніка в технологічних процесах»
ISSN 2219-9365

https://doi.org/10.31891/2219-9365-2024-79-7
UDC 004.056.5

TYMOSHCHUK Dmytro
Ternopil Ivan Puluj National Technical University
https://orcid.org/0000-0003-0246-2236
e-mail: dmytro.tymoshchuk@gmail.com
YATSKIV Vasyl
West Ukrainian National University
https://orcid.org/0000-0001-9778-6625
e-mail: jazkiv@ukr.net

USING HYPERVISORS TO CREATE A CYBER POLYGON

Cyber polygon used to train cybersecurity professionals, test new security technologies and simulate attacks play an important role in ensuring cybersecurity. The creation of such training grounds is based on the use of hypervisors, which allow efficient management of virtual machines, isolating operating systems and resources of a physical computer from virtual machines, ensuring a high level of security and stability. The paper analyses various aspects of using hypervisors in cyber polygons, including types of hypervisors, their main functions, and the specifics of their use in modelling cyber threats. The article shows the ability of hypervisors to increase the efficiency of hardware resources, create complex virtual environments for detailed modelling of network structures and simulation of real situations in cyberspace.
Keywords: Hypervisor, cyber polygon, VMware ESXi, Microsoft Hyper-V, Xen, KVM, Cluster.

ТИМОЩУК Дмитро
Тернопільський національний технічний університет імені Івана Пулюя
ЯЦКІВ Василь
Західноукраїнський національний університет

ВИКОРИСТАННЯ ГІПЕРВІЗОРІВ ДЛЯ СТВОРЕННЯ КІБЕРПОЛІГОНУ

Кіберполігони, які використовуються для тренування фахівців з кібербезпеки, тестування нових технологій захисту та моделювання атак, відіграють важливу роль у забезпечення кібербезпеки. В основі створення таких полігонів лежить використання гіпервізорів, які дозволяють ефективно управляти віртуальними машинами, ізолюючи операційні системи та ресурси фізичного комп'ютера від віртуальних машин, забезпечуючи високий рівень безпеки та стабільності. В роботі проаналізовано різні аспекти використання гіпервізорів у кіберполігонах, включаючи типи гіпервізорів, їхні основні функції, а також специфіку застосування у моделюванні кіберзагроз та розгортанні кіберполігонів. Показано здатність гіпервізорів підвищувати ефективність використання апаратних ресурсів, створювати складні віртуальні середовища для детального моделювання мережевих структур та імітації реальних ситуацій в кіберпросторі.
Ключові слова: Гіпервізор, кіберполігон, VMware ESXi, Microsoft Hyper-V, Xen, KVM, Cluster.

INTRODUCTION

Information technology is constantly evolving, creating new challenges for cybersecurity. Cyber polygon, or environments for modelling cyber threats, are becoming a necessary tool for training specialists, testing system security, and researching new methods of countering attacks. The basis for creating such training grounds is the use of hypervisors.

A hypervisor is software that manages the operation of virtual machines. It allows you to create and effectively manage virtual machines, each with its own operating system and applications.

A cyber polygon is a virtual environment used to train cybersecurity professionals, test new security technologies, simulate attacks, and practice incident response scenarios. It can include virtual machines, network equipment, storage systems and other components that simulate real-world infrastructure.

MAIN PART

The hypervisor distributes physical resources (such as CPU, memory, disk space, etc.) among several virtual machines, ensuring their independence and stability. This makes it possible to run several different operating systems on the same physical server, which increases the flexibility and efficiency of using hardware resources. The hypervisor ensures that virtual machines cannot interact with each other in an undesirable way or affect the host system, which provides a high level of security and stability. It can include various functions, such as managing virtual networks, data storage, and I/O processing. Thanks to its architecture, a hypervisor allows you to create and manage virtualised environments that can be used for a variety of purposes.

There are two types of hypervisors: type 1 (bare-metal) and type 2 (hosted).

Type 2 runs on top of an operating system that is already installed on a physical server (Figure 1). Examples: VMware Workstation, Oracle VirtualBox, Parallels Desktop, VMware Fusion, UTM, QEMU

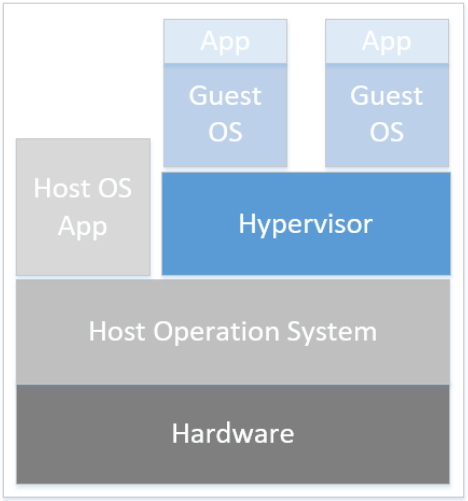

Fig. 1 Type 2 hypervisor architecture

Type 1 runs directly on the hardware without the need for an underlying operating system (Figure 2). Examples: VMware ESXi, Microsoft Hyper-V, Xen, and KVM.

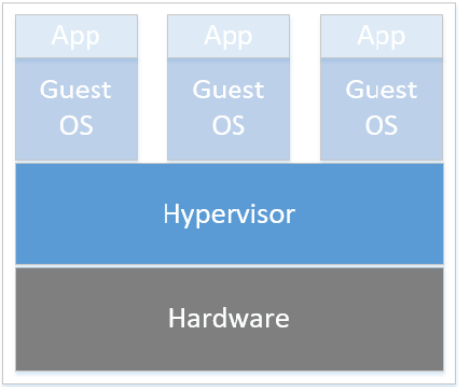

Fig. 2 Type 1 hypervisor architecture

VMware ESXi is a high-performance Type 1 hypervisor that is small and fast. The hypervisor delivers high performance while minimising virtualisation overheads thanks to its lightweight architecture [1]. High level of scalability. ESXi allows you to efficiently use server resources by running many virtual machines on a single server and automatically managing them through vCenter Server, allowing you to create large virtualised environments.

One of the main advantages of VMware ESXi is its high reliability. The hypervisor supports high availability features such as High Availability and Fault Tolerance. The vMotion function allows you to move virtual machines from one server to another without interrupting their operation, which is important for ensuring the smooth operation of services. The DRS (Distributed Resource Scheduler) tool automatically balances the load between servers by moving virtual machines according to available resources. Storage vMotion technology allows you to move virtual machines between different datastores without interrupting their operation. VMware vSAN software-defined storage system allows you to create virtualised storage based on local server disks. vSAN combines disc resources from multiple servers into a single cluster, creating scalable and high-performance storage for virtual machines. All of these are part of the VMware vSphere platform functionality and are critical technologies for ensuring the continuous operation of virtualised environments. VMware ESXi supports both Intel VT (Intel Virtualisation Technology) and AMD-V (AMD Virtualisation).

Intel VT and AMD-V are hardware-based virtualisation technologies developed by Intel and AMD, respectively. Intel VT and AMD-V are critical components for virtualisation performance in modern hypervisors. They leverage these hardware capabilities to improve virtualisation performance, security, and efficiency.

VMware ESXi can provide a solid foundation for a cyber polygon with its high-performance virtualisation capabilities, scalability, and reliability. It allows you to create complex and resource-intensive virtual environments that can mimic real-world network structures with different types of servers, operating systems, and applications. This allows you to configure various cyber threat and attack scenarios that can be tested without risking real systems. Using features such as vSphere vMotion and VMware vSAN, you can ensure that the cyber range is always running, even when performing updates or moving virtual machines between servers, which is important for long training and education sessions. VMware ESXi allows you to create complex network topologies with different network segments and access policies. This allows you to recreate real-world enterprise network conditions, including data distribution and protection. VMware ESXi also supports snapshots of virtual machines, which allows you to quickly return to previous system states after tests or attacks.

Hyper-V is a Type 1 hypervisor developed by Microsoft that allows you to create and manage virtual machines [2]. It is part of the Windows Server and Windows 10/11 (Pro, Enterprise, and Education) operating systems, providing users with the ability to efficiently virtualise servers and workstations. Hyper-V directly controls resources such as CPU, memory, and network interfaces, distributing them among virtual machines. Live Migration allows you to move running virtual machines between physical servers without interrupting their operation. Hyper-V Replica provides replication of virtual machines between two different sites, allowing for backup and disaster recovery. Despite its many benefits, Hyper-V does have some limitations. Although Hyper-V supports a variety of operating systems, some versions of Linux or other operating systems may have limited support or require additional configuration.

The Hyper-V hypervisor can serve as a solid foundation for a cyber polygon with its powerful virtualisation capabilities, integration with other Microsoft products, and flexibility in managing virtual environments. With Hyper-V, you can easily create isolated virtual environments in which to deploy a variety of operating systems and applications to simulate and defend against attacks. Integration with Active Directory allows you to control access to resources and manage users. Hyper-V also supports replication and backup functions, which allows you to preserve the state of your environment, quickly restore it after test attacks, or create backups for analysis. Hyper-V virtual switches allow you to configure complex network topologies by simulating various types of network interactions. With Hyper-V, you can implement a scalable cyber polygon that can be used for both individual training sessions and complex team exercises.

Xen is an open-source Type 1 hypervisor developed at the University of Cambridge and is one of the most popular virtualisation solutions in the world [3]. It allows you to run multiple operating systems on a single physical server, efficiently sharing hardware resources between them. Xen runs directly on the server hardware, providing a high level of performance and security by isolating virtual machines from each other.

One of the key features of Xen is its ability to support different virtualisation models, including paravirtualisation and hardware virtualisation. Paravirtualisation allows operating systems running on Xen to be optimised for virtualisation, which improves performance. Hardware virtualisation, on the other hand, allows you to run unmodified operating systems such as Windows through the use of special hardware extensions of the Intel VT or AMD-V processor. Xen supports high availability and live migration of virtual machines, allowing them to be moved between physical servers without interruption. Xen provides a command-line interface and tools for managing virtual machines, allowing administrators to effectively monitor and configure the virtualised environment.

Xen can be a good foundation for creating a cyber polygon due to its flexibility, open source, and ability to support different virtualisation models. Since Xen supports both paravirtualisation and hardware virtualisation, it allows you to create cyber polygons with different operating systems, including virtualisation-optimised versions of Linux and unmodified versions of Windows. Xen's flexibility allows you to create complex network topologies and scenarios to simulate real-world attacks. With support for live migration of virtual machines and high availability, Xen ensures the continuity of the cyber range, even during updates or the transfer of virtual machines to other physical servers. This allows you to conduct long-term training and testing without the risk of stopping the environment.

KVM (Kernel-based Virtual Machine) is a Type 1 hypervisor that is part of the Linux kernel and allows you to use hardware virtualisation features to create and manage virtual machines [4]. As a virtualisation solution, KVM delivers high performance and efficiency through tight integration with the Linux kernel, allowing it to use existing system components and resources. The main component of KVM is the kernel module, which turns the Linux kernel into a hypervisor that manages virtual machines.

One of the key advantages of KVM is that it uses standard Linux tools and mechanisms to manage virtual machines. This includes tools for monitoring, network configuration, and resource management. KVM supports a variety of operating systems, including different versions of Linux, Windows, and other OSes, making it a versatile virtualisation solution.

KVM integrates with other projects and technologies, such as libvirt, which provides interfaces for managing virtual machines via the command line or graphical interfaces.

Another important part of KVM is QEMU. QEMU (Quick Emulator) is a versatile open source emulator that allows you to create virtual machines by emulating various hardware [5].

QEMU provides emulation of a wide range of hardware components, such as processors, network cards, hard drives and other peripherals. One of the main advantages of QEMU is its ability to support different processor architectures such as x86, ARM and others. QEMU can run in both full emulation mode and hardware virtualisation mode to improve performance. In full emulation mode, QEMU provides emulation of all hardware, allowing you to run virtual machines regardless of whether the physical host supports hardware virtualisation.

KVM is an excellent basis for creating a cyber polygon due to its integration with the Linux kernel and powerful virtualisation capabilities. As a specialised environment for modelling and analysing cyber threats, a cyber polygon requires a stable, scalable and reliable platform for deploying virtual machines and simulating attacks. KVM, as part of the Linux kernel, provides high performance and efficiency through the use of hardware virtualisation via Intel VT or AMD-V. This allows you to create virtual machines with minimal overhead, which is especially important for large and resource-intensive environments such as cyber polygons. QEMU, when used in conjunction with KVM, provides the ability to emulate different hardware, allowing you to create virtual machines with different characteristics and configurations. This is useful for modelling various attack and defence scenarios within a cyber polygon. QEMU also supports virtual machine snapshots, which allows you to save system states and quickly restore them if necessary. The migration capability also helps to ensure high availability and scalability of the cyber polygon, adapting it to changing needs.

Type 1 hypervisors, which work directly with the hardware, provide higher performance than type 2 hypervisors because they do not require an underlying operating system. This is critical for cyber polygons, where high data processing speeds and realistic modelling of cyber threats are required. Due to these advantages, Type 1 hypervisors are ideal for creating a cyber polygon, providing a high level of performance, security, scalability and stability that are critical for effective testing and modelling of cyber threats.

One of the most important aspects of creating a cyber polygon is the use of routers with firewall functionality in the test environment. Routers such as the Cisco CSR1000v, IPFire, OPNsense, pfSense, MikroTik CHR, Juniper Cloud-Native Router, FortiGate VM, and Palo Alto VM-Series are a solid choice for a cyber polygon due to their advanced security, configuration, and integration with other systems [7][8][9][10][11][12][13]. Using these routers with firewall capabilities allows you to create a variety of scenarios for testing and training in the cyber lab. Their capabilities help to create complex and realistic network environments that allow you to effectively model and analyse cyber threats.

Figure 3 shows an architectural diagram of a cyber polygon created on the basis of a cluster of hypervisors.

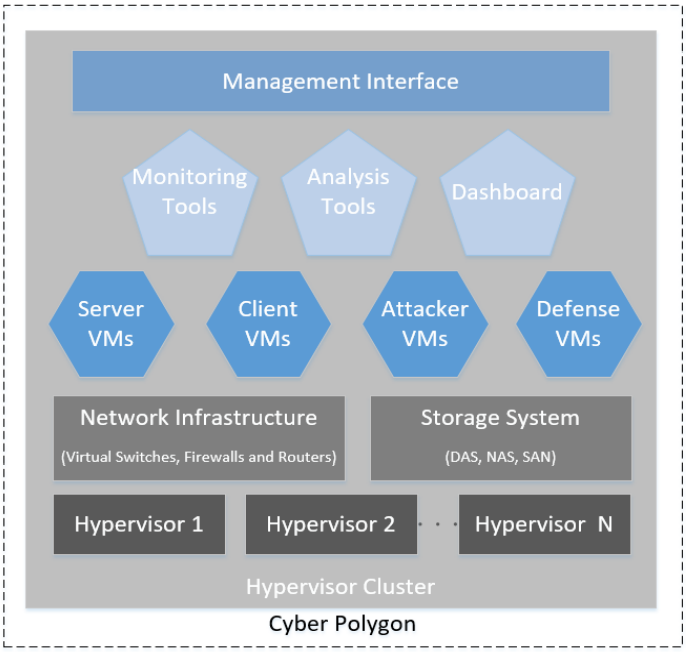

Fig. 3 The architecture of the cyber polygon

The centralised management interface allows you to manage all virtual machines, configure network configurations, and monitor the status of the entire hypervisor cluster. A hypervisor cluster consists of several physical servers with hypervisors. There are several types of virtual machines in the architecture: server machines that simulate various server roles, such as web servers, databases, file servers; client machines that simulate user endpoints, such as computers and mobile devices; attack machines configured with tools and scripts to perform cyber attacks; and machines that are equipped with security tools to detect and mitigate attacks. The network infrastructure includes virtual switches, routers and firewalls that control and monitor data flows between virtual machines and separate network segments. The storage system includes DAS (Direct-Attached Storage), NAS (Network-Attached Storage), and SAN (Storage Area Network), which provide data storage for virtual machines. Monitoring, analysis tools and a dashboard allow real-time monitoring and analysis of network activity and incident response by visualising the security status.

This architecture allows for the creation of diverse and realistic scenarios for cybersecurity training, providing the flexibility and scalability to conduct complex tests in a controlled environment.

CONCLUSIONS

Using a Type 1 hypervisor to create a cyber polygon is an effective approach that allows you to simulate complex and realistic cyber threat scenarios. The main advantages are high performance, reliability and isolation of virtual machines, which creates a secure environment for testing and training. Type 1 hypervisors provide the ability to create scalable and adaptive cyber polygons that can be used to train cybersecurity professionals, conduct research and test new security technologies.

Thus, a Type 1 hypervisor is a key element for creating effective and secure cyber polygons that can be used in various industries, from educational institutions to corporate environments.

References

1. VMware vSphere Documentation. URL: https://docs.vmware.com/en/VMware-vSphere/index.html
2. Hyper-V technology overview. URL: https://learn.microsoft.com/en-us/windows-server/virtualization/hyper-v/hyper-v-technology-overview
3. Xen project software overview. URL: https://wiki.xenproject.org/wiki/Xen_Project_Software_Overview#PV_.28x86.29
4. Kernel Virtual Machine. URL: https://linux-kvm.org/page/Main_Page
5. QEMU documentation. URL: https://www.qemu.org/docs/master/
6. Introduction – OPNsense documentation. URL: https://docs.opnsense.org/intro.html
7. IPFire documentation. URL: https://www.ipfire.org/docs
8. Introduction pfSense Documentation. Netgate Documentation. URL: https://docs.netgate.com/pfsense/en/latest/general/index.html
9. Cisco Cloud Services Router 1000v. URL: https://www.cisco.com/c/en/us/support/routers/cloud-services-router-1000v/model.html
10. Cloud hosted router - mikrotik documentation. URL: https://help.mikrotik.com/docs/display/ROS/Cloud+Hosted+Router,+CHR
11. Juniper Cloud-Native Router (JCNR). URL: https://www.juniper.net/documentation/product/us/en/juniper-cloud-native-router/
12. FortiGate private cloud 7.4. Fortinet Document Library. URL: https://docs.fortinet.com/product/fortigate-private-cloud/7.4
13. VM-Series. Palo Alto Networks Tech Docs Home. URL: https://docs.paloaltonetworks.com/vm-series

August 16, 2024;
Oxford, United Kingdom

SECTION 11.
INFORMATION TECHNOLOGIES AND SYSTEMS

DOI 10.36074/logos-16.08.2024.036

SLOWLORIS DDOS DETECTION AND PREVENTION IN REAL-TIME

Dmytro Tymoshchuk[1], Vasyl Yatskiv[2]

1. *Ternopil Ivan Puluj National Technical University, UKRAINE*
ORCID ID: 0000-0003-0246-2236

2. *West Ukrainian National University, UKRAINE*
ORCID ID: 0000-0001-9778-6625

Abstract. *Web servers play a key role in providing access to online resources. However, they are targeted by a variety of cyber attacks, including DDoS attacks such as Slowloris, which can paralyse servers by draining their resources. This paper investigates methods for detecting and preventing Slowloris attacks in real time. The main focus was on algorithms and tools that can be used to protect web servers. The architecture of streaming and asynchronous web servers, their vulnerability to Slowloris attacks, and recommendations for improving server configuration to minimise risks were analysed. Software has been developed that monitors traffic in real time, detects suspicious activity, and automatically blocks attacking IP addresses using a firewall. The functionality of notifications about attacks via Telegram Bot and logging of events for further analysis has been implemented. The obtained results showed that the developed measures can effectively protect servers from Slowloris attacks, ensuring their uninterrupted operation.*

Introduction. Internet technologies play a key role in the functioning of various service industries. However, as the volume of online traffic grows, so does the threat to web servers in the form of cyber attacks. One of these attacks is Slowloris, a type of distributed denial of service (DDoS) attack that is designed to exhaust a web server's resources, making it unavailable to legitimate users [1]. Detecting and preventing Slowloris attacks in real time is critical to protecting network infrastructure and ensuring the smooth operation of websites. This paper discusses methods for detecting and preventing real-time Slowloris DDoS attacks, with a particular focus on algorithms and tools that can be used to protect servers.

Main part. Web servers are the main components that provide access to web pages on the World Wide Web. They can serve many requests simultaneously from

different clients, which provides fast access to content for many users at the same time.

Web servers can be divided into two main types of architecture: streaming and asynchronous.

Apache is a streaming web server [2]. Streaming web servers use a request processing model where each request is processed by a separate thread or process. When a client sends a request to a streaming web server, the server creates a separate thread or process to handle that request. This means that each request is processed separately from other requests. In streaming servers, each thread or process is blocked while the request is being processed. This can lead to the entire server being blocked in the event of a large number of requests. Streaming servers often allow developers to use synchronous code, which makes application development easier. Developers don't have to worry about handling competing processes or threads.

Nginx is an example of an asynchronous server [3]. Asynchronous web servers use a request processing model where a single thread or process can serve many requests without blocking. Nginx uses an approach with an asynchronous, non-blocking, event-driven architecture. This allows the web server to handle multiple connections within a single process. Nginx has a main process that performs privileged operations such as port binding, reading and evaluating configuration files, and creating child processes.

Due to its asynchronous architecture, the server spends fewer resources on creating and managing threads or processes, which allows it to be more productive and faster. Asynchronous servers use a non-blocking processing model, which allows the server to process requests without waiting for previous operations to complete. Nginx is ideal for serving many concurrent connections, making it popular for load balancing tasks and handling large volumes of traffic.

For all its capabilities, Apache is vulnerable to Slowloris attacks in its streaming architecture. The Nginx web server has a different architecture and, if configured appropriately, is not vulnerable to Slowloris attacks.

The principle of the Slowloris attack is that the attacker tries to keep a large number of open connections to the web server, which leads to the server allocating resources to each open connection. This causes the web server to be unavailable to legitimate users, as all available resources are directed to serve the attackers. One of the key points of the Slowloris attack is request delay, where the attacker tries to delay sending full requests by opening a connection but not completing it completely. This forces the server to wait for the request to complete and keep the connection open.

Figure 1 shows the correct interaction between the client and the web server.

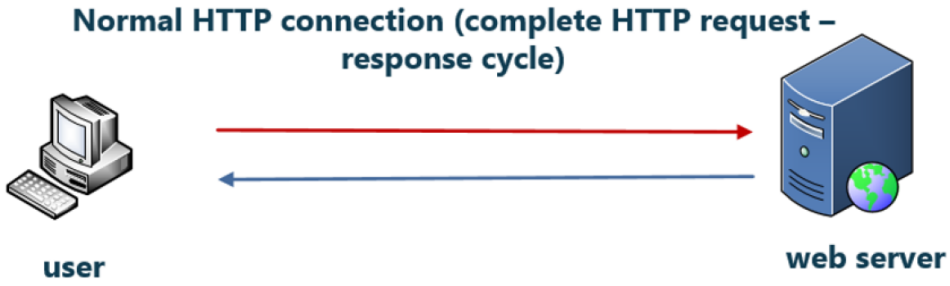

Fig. 1. **Correct HTTP connection process**

The user sends an HTTP request to the web server. The web server processes the request and sends back an HTTP response. This cycle represents the fundamental communication mechanism used in web browsing and data exchange over the Internet.

Slowloris works in a different mode, using the "Keep-Alive" header of the HTTP request, as in a correct interaction. This header informs the server that there will be additional requests in the future, so an attacker can send several incomplete requests and keep them open. This causes the server to reach its connection limit and cannot process new requests from legitimate users. The attacker then sends additional incomplete requests to maintain control over existing connections and prevent legitimate users from accessing the server. By sending enough data with each request, but not completing it, the attackers effectively tie up the web server's resources for a long period of time (Fig. 2).

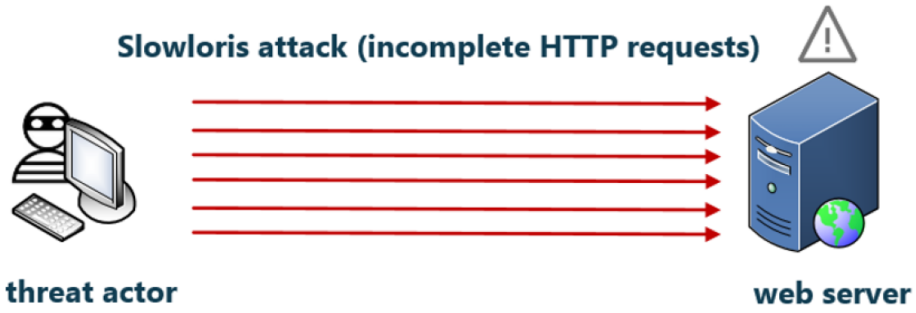

Fig. 2. **Principles of the Slowloris attack on a web server**

Attackers can use a botnet or other network resources to send multiple requests to a web server at the same time (Fig. 3).

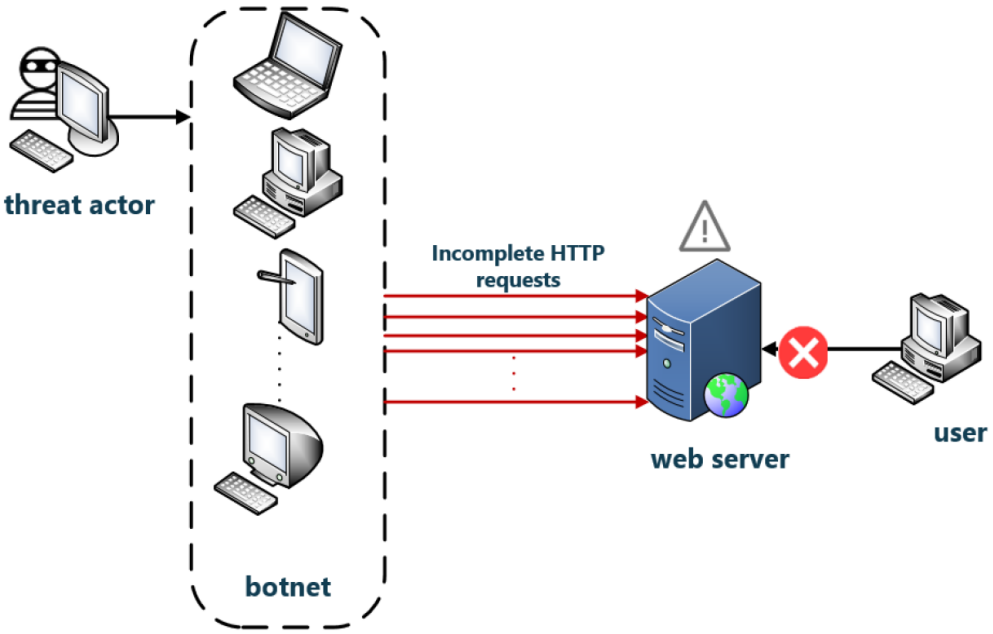

Fig. 3. **Principles of the Slowloris attack using a botnet**

This allows you to increase the strength of the attack and its ability to overwhelm the server. The attacker gradually drains the server's resources, and it can be extremely effective with a large number of simultaneous connections.

Using IoT devices for a Slowloris attack is particularly dangerous given the growing number of such devices and their limited security. Attackers can seize control of numerous IoT devices, combining them into a botnet. This botnet can then be used to launch Slowloris attacks against web servers. Due to the large number of devices, the attack can be extremely effective, even if each individual device only sends a small number of requests.

Since the Slowloris attack is characterised by a low volume of traffic sent by attackers, it is difficult to detect.

One of the obvious results of a Slowloris attack is the temporary or prolonged unavailability of a web server to legitimate users. The server may be overloaded and unable to respond to requests. This attack can lead to significant losses for the business served by the web server and can damage the reputation of the organisation.

Defence against Slowloris attacks includes a variety of techniques and approaches aimed at reducing the risk and minimising the consequences of this attack.

One of the first steps in protecting against Slowloris is to improve the configuration of the web server, in particular, to increase the maximum number of simultaneous connections that the server can handle. But you need to keep in mind that the server's capabilities are not unlimited and can also be exhausted. It is also important to set up timeouts for terminating inactive connections.

Web servers have modules or plugins that are designed with the previous parameters in mind to reduce the impact of a Slowloris attack. However, when an attack is launched from multiple sources, using Apache modules that can mitigate the attack does not have the desired effect. Because mod_reqtimeout effectively rejects slow connections, but it does not prevent these connections from being made. This means that during the mod_reqtimeout grace period, Apache connection slots can be filled. mod_limitipconn is triggered too late and cannot mitigate Slowloris. It is designed to return an HTTP 503 to the client if the client is making too many connections, but assumes that the client wants a response. Slowloris, of course, does not need this, so the connection slots remain busy.

With a reverse proxy server, you can create an additional barrier between clients and the web server, which reduces the server's vulnerability to a Slowloris attack. In addition, a reverse proxy can improve the overall security and efficiency of the server by providing protection against other threats.

Another way to reduce the impact of the attack is to use firewalls to block IP addresses from which Slowloris attacks are launched. The development of an algorithm, methodology for this process and its automation can help in the effective detection and blocking of attackers.

The development of a Slowloris attack detection algorithm and protection software requires a comprehensive approach and integration with different layers of server infrastructure. Since the Slowloris attack involves the use of many incomplete HTTP requests to overload the target server, real-time monitoring of TCP ports 80 and 443 will help detect the attack.

The developed software identifies the IP addresses that initiate the attack and blocks them by using a firewall. This service constantly monitors and analyses incoming traffic, and if signs of Slowloris are detected, it automatically adds the IP addresses to the blocked list. The developed software has the ability to notify about the detection of an attack using Telegram Bot, which makes it possible to promptly inform about the attack (Fig. 4).

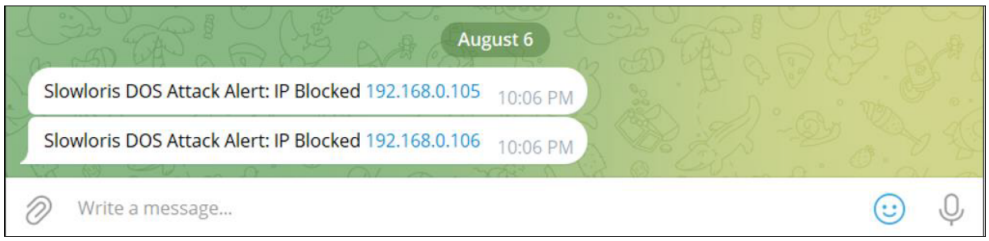

Fig 4. **Telegram message about the Slowloris attack**

The development of the functionality to run the developed software protection as a service made it possible to automatically enable it when the server starts up and constantly monitor traffic. The software keeps an event log, recording details about each blocked IP address (Fig. 5).

Fig 5. **Contents of the /var/log/doswww.log log file**

This log file can be used in SIEM to track attacks, detect security threats, and respond to network events..

Conclusions. The Slowloris DDoS attack is a serious threat to modern web servers, but it can be effectively detected and blocked using modern methods and tools. The advanced algorithm and practical implementation of software protection made it possible to protect the Apache web server from Slowloris attacks on the Linux platform. The developed protection includes integration with the firewall to block the attack in real time, an attack logging system and the ability to receive notifications on Telegram Bot about the incident.

REFERENCES:

[1] Slowloris DDoS attack. (n. d.). https://httpd.apache.org/docs/2.4/developer/
[2] Developer documentation for the apache HTTP server 2.4 - apache HTTP server version 2.4. (n. d.). https://httpd.apache.org/docs/2.4/developer/
[3] Inside NGINX: How we designed for performance & scale – NGINX community blog. (n. d.). https://www.nginx.com/blog/inside-nginx-how-we-designed-for-performance-scale/

ДОДАТОК Б. Лог файл сервера Apache access.log

```
192.168.0.105 - - [02/Aug/2024:12:43:55 +0300] "GET /?1649 HTTP/1.1" 408 482 "-" "Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.0"
192.168.0.105 - - [02/Aug/2024:12:43:55 +0300] "GET /?456 HTTP/1.1" 408 482 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko"
192.168.0.105 - - [02/Aug/2024:12:43:55 +0300] "GET /?1701 HTTP/1.1" 408 482 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko"
192.168.0.105 - - [02/Aug/2024:12:43:55 +0300] "GET /?1808 HTTP/1.1" 408 482 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko"
192.168.0.105 - - [02/Aug/2024:12:43:55 +0300] "GET /?16 HTTP/1.1" 408 482 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko"
192.168.0.105 - - [02/Aug/2024:12:43:55 +0300] "GET /?640 HTTP/1.1" 408 482 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:49.0) Gecko/20100101 Firefox/49.0"
192.168.0.105 - - [02/Aug/2024:12:43:55 +0300] "GET /?1822 HTTP/1.1" 408 482 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:49.0) Gecko/20100101 Firefox/49.0"
192.168.0.105 - - [02/Aug/2024:12:43:55 +0300] "GET /?1507 HTTP/1.1" 408 482 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:49.0) Gecko/20100101 Firefox/49.0"
192.168.0.105 - - [02/Aug/2024:12:43:55 +0300] "GET /?1865 HTTP/1.1" 408 482 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:49.0) Gecko/20100101 Firefox/49.0"
192.168.0.105 - - [02/Aug/2024:12:43:55 +0300] "GET /?378 HTTP/1.1" 408 482 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64; rv:49.0) Gecko/20100101 Firefox/49.0"
192.168.0.105 - - [02/Aug/2024:12:43:55 +0300] "GET /?683 HTTP/1.1" 408 482 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:49.0) Gecko/20100101 Firefox/49.0"
```

192.168.0.105 - - [02/Aug/2024:12:43:55 +0300] "GET /?1821 HTTP/1.1" 408 482 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:49.0) Gecko/20100101 Firefox/49.0"
192.168.0.105 - - [02/Aug/2024:12:43:55 +0300] "GET /?338 HTTP/1.1" 408 482 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:49.0) Gecko/20100101 Firefox/49.0"
192.168.0.105 - - [02/Aug/2024:12:43:55 +0300] "GET /?284 HTTP/1.1" 408 482 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:49.0) Gecko/20100101 Firefox/49.0"
192.168.0.105 - - [02/Aug/2024:12:43:55 +0300] "GET /?1228 HTTP/1.1" 408 482 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:49.0) Gecko/20100101 Firefox/49.0"
192.168.0.105 - - [02/Aug/2024:12:43:55 +0300] "GET /?1580 HTTP/1.1" 408 482 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36"
192.168.0.105 - - [02/Aug/2024:12:43:55 +0300] "GET /?446 HTTP/1.1" 408 482 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36"
192.168.0.105 - - [02/Aug/2024:12:43:55 +0300] "GET /?1403 HTTP/1.1" 408 482 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36"
192.168.0.105 - - [02/Aug/2024:12:43:55 +0300] "GET /?257 HTTP/1.1" 408 482 "-" "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36"
192.168.0.105 - - [02/Aug/2024:12:43:55 +0300] "GET /?1250 HTTP/1.1" 408 482 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36"
192.168.0.105 - - [02/Aug/2024:12:43:55 +0300] "GET /?1770 HTTP/1.1" 408 482 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.79 Safari/537.36 Edge/14.14393"

192.168.0.105 - - [02/Aug/2024:12:43:55 +0300] "GET /?870 HTTP/1.1" 408 482 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.79 Safari/537.36 Edge/14.14393"
192.168.0.105 - - [02/Aug/2024:12:43:55 +0300] "GET /?1292 HTTP/1.1" 408 482 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.79 Safari/537.36 Edge/14.14393"
192.168.2.10 - - [02/Aug/2024:13:11:50 +0300] "GET / HTTP/1.1" 200 5649 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:20 +0300] "GET / HTTP/1.1" 200 5649 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:20 +0300] "GET /assets/plugins/jquery-scrollTo/jquery.scrollTo.min.js HTTP/1.1" 200 1643 "http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:20 +0300] "GET /assets/plugins/jquery-1.12.3.min.js HTTP/1.1" 200 34141 "http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:20 +0300] "GET /assets/css/styles.css HTTP/1.1" 200 3667 "http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:20 +0300] "GET /assets/plugins/font-awesome/css/font-awesome.css HTTP/1.1" 200 7370 "http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:20 +0300] "GET /assets/plugins/bootstrap/js/bootstrap.min.js HTTP/1.1" 200 10178 "http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:20 +0300] "GET /assets/plugins/bootstrap/css/bootstrap.min.css HTTP/1.1" 200

```
20083 "http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0;
Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET
/assets/js/main.js HTTP/1.1" 200 735 "http://192.168.1.100/"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET
/assets/images/feature-3.png HTTP/1.1" 200 102732
"http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64;
x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET
/assets/images/feature-2.png HTTP/1.1" 200 238570
"http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64;
x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET
/assets/images/feature-7.png HTTP/1.1" 200 109929
"http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64;
x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET
/assets/images/feature-5.png HTTP/1.1" 200 119153
"http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64;
x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET
/assets/images/feature-6.png HTTP/1.1" 200 95378
"http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64;
x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET
/assets/images/imac.png HTTP/1.1" 200 118350
"http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64;
x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET
/assets/images/feature-4.png HTTP/1.1" 200 146811
"http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64;
x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET
/assets/images/logo-css3.svg HTTP/1.1" 200 1976
```

"http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET /assets/images/logo-less.svg HTTP/1.1" 200 11085 "http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET /assets/images/logo-html5.svg HTTP/1.1" 200 2358 "http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET /assets/images/logo-bootstrap.svg HTTP/1.1" 200 3597 "http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET /assets/images/logo-icon.svg HTTP/1.1" 200 1307 "http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET /assets/images/hero-1.jpg HTTP/1.1" 200 99274 "http://192.168.1.100/assets/css/styles.css" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET /assets/images/logo-angular.svg HTTP/1.1" 200 2591 "http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET /assets/images/map.png HTTP/1.1" 200 75801 "http://192.168.1.100/assets/css/styles.css" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET /assets/images/profile-1.png HTTP/1.1" 200 26799 "http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"

192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET /assets/plugins/font-awesome/fonts/fontawesome-webfont.woff2?v=4.6.3 HTTP/1.1" 200 72185 "http://192.168.1.100/assets/plugins/font-awesome/css/font-awesome.css" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET /assets/images/profile-2.png HTTP/1.1" 200 23538 "http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET /assets/images/figure-2.png HTTP/1.1" 200 8019 "http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET /assets/images/logo-jquery.svg HTTP/1.1" 200 28896 "http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET /assets/images/figure-3.png HTTP/1.1" 200 4506 "http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET /assets/images/profile-3.png HTTP/1.1" 200 25265 "http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET /assets/images/figure-1.png HTTP/1.1" 200 12847 "http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET /assets/images/profile-4.png HTTP/1.1" 200 24968 "http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET /assets/images/feature-8.png HTTP/1.1" 200 112806

"http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET /assets/images/feature-1.png HTTP/1.1" 200 106981 "http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET /assets/images/team-1.png HTTP/1.1" 200 36067 "http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET /assets/images/team-2.png HTTP/1.1" 200 63212 "http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:21 +0300] "GET /favicon.ico HTTP/1.1" 200 1450 "http://192.168.1.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"
192.168.2.10 - - [02/Aug/2024:13:12:32 +0300] "GET /assets/images/hero-2.jpg HTTP/1.1" 200 126262 "http://192.168.1.100/assets/css/styles.css" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0"

ДОДАТОК В. Лістинг файлу alertdoswww.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>


#define MAX_LINE_LENGTH 256

// Declaration of the function for logging messages to a file
void logMessage(const char *log_filename, const char *message);

// Function to send a notification to Telegram
void sendNotification(const char *ip);

// Function to read configuration values from a file
int readConfig(const char *filename, int *BLOCK_THRESHOLD, int
*MAX_IP_COUNT) {
    FILE *config_file = fopen(filename, "r");
    if (config_file == NULL) {
        perror("Failed to open configuration file");
        return 1;
    }

    char line[MAX_LINE_LENGTH];

    while (fgets(line, sizeof(line), config_file) != NULL) {
        if (strstr(line, "BLOCK_THRESHOLD=") != NULL) {
            sscanf(line, "BLOCK_THRESHOLD=%d", BLOCK_THRESHOLD);
        } else if (strstr(line, "MAX_IP_COUNT=") != NULL) {
            sscanf(line, "MAX_IP_COUNT=%d", MAX_IP_COUNT);
        }
    }
```

```
    fclose(config_file);
    return 0;
}

int main() {
    int BLOCK_THRESHOLD = 0;
    int MAX_IP_COUNT = 0;

    if      (readConfig("/etc/doswww.conf",      &BLOCK_THRESHOLD,
&MAX_IP_COUNT) != 0) {
        return 1;
    }

    FILE *file = popen("netstat -an", "r");
    if (file == NULL) {
        perror("Failed to run netstat");
        return 1;
    }

    char line[MAX_LINE_LENGTH];

    // Arrays to store unique IP addresses and their counts
    char *unique_ips[MAX_IP_COUNT];
    int ip_counts[MAX_IP_COUNT];
    int ip_blocked[MAX_IP_COUNT];
    for (int i = 0; i < MAX_IP_COUNT; i++) {
        unique_ips[i]  =  NULL;  //  Initialize  array  with  NULL
pointers
        ip_counts[i] = 0;
        ip_blocked[i] = 0;  // Initialize blocked flag to 0
    }

    while (fgets(line, sizeof(line), file) != NULL) {
        //  Check  if  the  line  contains  the  desired  ports  and
"ESTABLISHED"
```

```
        if ((strstr(line, ":443") != NULL || strstr(line, ":80")
!= NULL) && strstr(line, "ESTABLISHED") != NULL) {
            char *token = strtok(line, " ");
            int i = 0;
            while (token != NULL) {
                if (i == 4) { // Check for the fifth column
(remote address)
                    // Assuming the IP address is in the format
IP:PORT
                    char *ip_port = strtok(token, ":");
                    int existing_ip_index = -1;
                    for (int j = 0; j < MAX_IP_COUNT; j++) {
                        if    (unique_ips[j]    !=    NULL    &&
strcmp(unique_ips[j], ip_port) == 0) {
                            existing_ip_index = j;
                            break;
                        }
                    }
                    if (existing_ip_index == -1) {
                        // New unique IP address
                        for (int j = 0; j < MAX_IP_COUNT; j++) {
                            if (unique_ips[j] == NULL) {
                                unique_ips[j] = strdup(ip_port);
                                ip_counts[j] = 1;
                                ip_blocked[j]    =    0;       //
Initialize blocked flag to 0
                                break;
                            }
                        }
                    } else if (!ip_blocked[existing_ip_index]) {
// Check if IP hasn't been blocked
                        if    (ip_counts[existing_ip_index]    <=
BLOCK_THRESHOLD) {
                            //  IP  address  hasn't  exceeded  the
threshold
                            ip_counts[existing_ip_index]++;
```

```
                        }
                        if     (ip_counts[existing_ip_index]     >
BLOCK_THRESHOLD) {
                            // Block the IP address and log the
message
                            char command[MAX_LINE_LENGTH];
                            snprintf(command,    sizeof(command),
"sudo    iptables    -A    INPUT    -s    %s    -j    DROP",
unique_ips[existing_ip_index]);
                            system(command);
                            printf("Blocked      IP:      %s\n",
unique_ips[existing_ip_index]);

                            // Log the blocked IP address along
with the current date and time
                            char log_message[MAX_LINE_LENGTH];
                            time_t now;
                            struct tm *tm_info;
                            time(&now);
                            tm_info = localtime(&now);
                            strftime(log_message,
MAX_LINE_LENGTH, "%Y-%m-%d %H:%M:%S", tm_info);
                            snprintf(log_message                +
strlen(log_message),  MAX_LINE_LENGTH  -  strlen(log_message),  "
Slowloris    DOS    Attack    Alert:    IP    Blocked    %s",
unique_ips[existing_ip_index]);
                            logMessage("/var/log/doswww.log",
log_message);

                            // Mark IP as blocked
                            ip_blocked[existing_ip_index] = 1;

                            // Send a notification to Telegram

sendNotification(unique_ips[existing_ip_index]);
                        }
```

```
                    }
                }
                token = strtok(NULL, " ");
                i++;
            }
        }
    }

    pclose(file);

    // Print unique IP addresses and their counts
    for (int i = 0; i < MAX_IP_COUNT && unique_ips[i] != NULL;
i++) {
        printf("Unique IP: %s, Count: %d\n", unique_ips[i],
ip_counts[i]);
    }

    return 0;
}

// Function for logging messages to a file
void logMessage(const char *log_filename, const char *message) {
    FILE *log_file = fopen(log_filename, "a");
    if (log_file != NULL) {
        fprintf(log_file, "%s\n", message);
        fclose(log_file);
    }

}

// Function to send a notification to Telegram
void sendNotification(const char *ip) {
    char curl_command[MAX_LINE_LENGTH];
    snprintf(curl_command, sizeof(curl_command), "curl -s -X
POST
https://api.telegram.org/botAAAAAAAAAA:XXXXXXXXXXXXXXXXXXXXX_YYY
```

```
YYYYYYYYYYYYYYYYYYYY/sendMessage -d chat_id=XXXXXX -d
text=\"Slowloris DOS Attack Alert: IP Blocked %s\" &>/dev/null
2>&1", ip);
    system(curl_command);
}
```