Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(освітній рівень)

на тему: "Використання мови Lua для розширення функціональних можливостей IDS/IPS"

Виконав: студент (ка)
Спеціальності:
125 «Кібербезпека та захист інформації»
(шифр і назва напряму підготовки, спеціальності)
Чурбаков К.О.
підпис (прізвище та ініціали)

Керівник Карпінський М.П..
підпис (прізвище та ініціали)

Нормоконтроль Тимощук Д. І.
підпис (прізвище та ініціали)

Завідувач кафедри Загородна Н.В.
підпис (прізвище та ініціали)

Рецензент Жаровський Р.О.
підпис (прізвище та ініціали)

м. Тернопіль – 2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра Кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
«____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека та захист інформації
(шифр і назва спеціальності)

Студенту Чурбакову Константину Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Використання мови Lua для розширення функціональних можливостей IDS/IPS

Керівник роботи Карпінський Микола Петрович, д.т.н., професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «20» 11 2024 року № 4/7-1112

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи Наукові публікації про використання IDS/IPS та основи, переваги та можливості інтеграції мови Lua

4. Зміст роботи (перелік питань, які потрібно розробити): Вступ, 1. Аналіз сучасного стану технологій IDS/IPS, 1.1 Загальний огляд систем виявлення та запобігання вторгнень (IDS/IPS), 1.2 Переваги та недоліки використання скриптових мов у IDS/IPS, 1.3 Використання машинного навчання та штучного інтелекту для підвищення ефективності IDS/IPS, 2. Дослідження теоретичних основ розширення IDS/IPS за допомогою Lua, 2.1 Основи мови Lua: переваги, можливості інтеграції та особливості використання, 2.2 Огляд механізмів розширення Suricata за домогою Lua-скриптів, 2.3 Методологічні основи інтеграції Lua З іншими компонентами кібербезпеки (Threat Intelligence, SIEM та SOAR), 3. Розробка практичного рішення розширення IDS/IPS за допомогою Lua, 3.1 Встановка та конфігурація SuriCata 3.2 Розробка та тестування Lua-скриптів для виявлення загроз, 4.Охорона праці та безпека в надзвичайних ситуаціях, 4.1 Охорона праці, 4.2 Безпека в надзвичайних ситуаціях

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Тема роботи, виконавець та науковий керівник, 2. Наукова новизна дослідження, 3. Актуальність теми роботи, 4. Мета та завдання дослідження, 5. Аналіз сучасного стану технологій IDS/IPS, 6. Дослідження можливостей розширення IDS/IPS за допомогою Lua, 7. Практичне впровадження Lua-скриптів у платформу Suricata, 8. Результати тестування та аналіз ефективності, 9. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Осухівська Г.М., к.т.н., доцент		
Безпека в надзвичайних ситуаціях	Клепчик В.М., проректор з адміністративно-господарської роботи та будівництва		

7. Дата видачі завдання 10.12.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	24.09	*Виконано*
2.	Підбір наукових джерел про системи IDS/IPS	25.09 – 29.09	*Виконано*
3.	Аналіз наукових джерел про використання мови Lua в системах IDS/IPS	01.10	*Виконано*
4.	Виконання огляду мови Lua та її можливостей інтеграції	02.10–06.10	*Виконано*
5.	Дослідження методів розширення функціональності Suricata за допомогою Lua	07.10–10.10	*Виконано*
6.	Розробка Lua-скриптів для виявлення загроз у мережевому трафіку	11.10–15.10	*Виконано*
7.	Конфігурація та налаштування системи Suricata для підтримки Lua	16.10–20.10	*Виконано*
8.	Проведення тестування розроблених Lua-скриптів	21.10–25.10	*Виконано*
9.	Аналіз результатів тестування для оцінки ефективності роботи скриптів	26.10–30.10	*Виконано*
10.	Оформлення розділу «Охорона праці та безпека в надзвичайних ситуаціях»	11.12–12.12	*Виконано*
11.	Підготовка звіту та оформлення кваліфікаційної роботи	13.12–14.12	*Виконано*
12.	Нормоконтроль	16.12	*Виконано*
13.	Перевірка на плагіат	17.12	*Виконано*
14.	Попередній захист кваліфікаційної роботи		*Виконано*
15.	Захист кваліфікаційної роботи		

Студент _____ Чурбаков К.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Карпінський М.П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Використання мови Lua для розширення функціональних можливостей IDS/IPS // ОР «Магістр» // Чурбаков Константин Олексійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБм-62 // Тернопіль, 2024 // С. 81, рис. – 21, табл. – 8, додат. – 3.

Ключові слова: Suricata, Lua, IDS, IPS, портсканування, кібербезпека, скрипти, аналіз трафіку, виявлення загроз, інтеграція, мережевий захист.

Метою роботи є дослідження можливостей розширення функціональності систем IDS/IPS за допомогою мови Lua та створення користувацьких алгоритмів для Suricata.

У роботі проведено аналіз IDS/IPS систем, розроблено та протестовано Lua-скрипти для виявлення порт-сканування, що дозволяє адаптувати систему до нових кіберзагроз.

Розглянуто сучасний стан IDS/IPS технологій, досліджено інтеграцію мови Lua з Suricata, а також детально описано розробку і тестування скриптів для виявлення загроз.

Окрему увагу приділено питанням охорони праці, безпеки та впливу здорового способу життя на професійну діяльність.

ANNOTATION

Using Lua to Extend the Functionality of IDS/IPS // Thesis of educational level "Master"// // Churbakov Konstantyn // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cybersecurity, group СБм-62 // Ternopil, 2024 // P. 81, figs. 21, tables 8, appendix 3.

Keywords: Suricata, Lua, IDS, IPS, port scanning, cybersecurity, scripts, traffic analysis, threat detection, integration, network protection.

The aim of this work is to study the possibilities of extending the functionality of IDS/IPS systems using the Lua language and creating custom algorithms for Suricata.

The paper analyzes IDS/IPS systems, develops and tests Lua scripts for detecting port scanning, which allows the system to adapt to new cyber threats.

The current state of IDS/IPS technologies is reviewed, the integration of Lua with Suricata is investigated, and the development and testing of threat detection scripts is described in detail.

Particular attention is paid to the issues of occupational health and safety and the impact of a healthy lifestyle on professional activities.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IDS — Intrusion Detection System

IPS — Intrusion Prevention System

TCP — Transmission Control Protocol

UDP — User Datagram Protocol

HTTP — Hypertext Transfer Protocol

JSON — JavaScript Object Notation

SIEM — Security Information and Event Management

SOAR — Security Orchestration, Automation, and Response

RFC — Request for Comments

DNS — Domain Name System

DDoS — Distributed Denial of Service

IoT — Internet of Things

ICS — Industrial Control Systems

SCADA — Supervisory Control and Data Acquisition

5G — Fifth-Generation Networks

SDN — Software-Defined Networking

TLS 1.3 — Transport Layer Security Version 1.3

SSL — Secure Sockets Layer

DPI — Deep Packet Inspection

PCI DSS — Payment Card Industry Data Security Standard

GDPR — General Data Protection Regulation

HIPAA — Health Insurance Portability and Accountability Act

ML – Machine Learning

AI - Artificial Intelligence

ВСТУП

У сучасному світі інформаційна безпека є однією з ключових складових функціонування будь-якої організації, незалежно від її масштабу чи сфери діяльності. З кожним роком кількість кіберзагроз і складність атак зростають, створюючи значні виклики для фахівців у галузі захисту інформації. Ефективним способом протидії сучасним загрозам є використання систем виявлення та запобігання вторгнень (IDS/IPS). Такі системи забезпечують аналіз мережевого трафіку, виявлення потенційних загроз та швидку реакцію на атаки. Проте в умовах еволюції методів атак традиційні підходи до аналізу можуть бути недостатньо ефективними.

Одним зі способів підвищення ефективності IDS/IPS є інтеграція сучасних технологій програмування, зокрема використання мови Lua. Lua є легкою, високопродуктивною скриптовою мовою, яка дозволяє розширювати функціональні можливості систем без необхідності змінювати їхній основний код. Її простота, гнучкість і здатність інтегруватися у різні платформи роблять Lua ідеальним вибором для створення користувацьких правил виявлення, обробки специфічного трафіку чи впровадження нових алгоритмів аналізу даних.

Сучасні IDS/IPS, такі як Suricata, активно впроваджують підтримку мови Lua, що відкриває можливості для створення гнучких, адаптивних і високоефективних рішень. Зокрема, Suricata дозволяє використовувати Lua для глибокого аналізу трафіку, створення складних правил виявлення загроз, а також інтеграції з іншими системами кіберзахисту.

Метою даної роботи є дослідження можливостей використання мови Lua для розширення функціональних можливостей систем виявлення та запобігання вторгнень на прикладі платформи Suricata. У рамках роботи здійснюється розробка Lua-скриптів для вирішення завдань виявлення мережевих загроз, таких як порт-сканування, шляхом впровадження

користувацьких алгоритмів і правил. Це забезпечує можливість адаптації системи до сучасних кіберзагроз.

Для досягнення поставленої мети було проведено аналіз сучасних підходів до інтеграції Lua у платформу Suricata, розроблено Lua-скрипти для ідентифікації складних загроз та протестовано їх у реальному середовищі.

Наукова новизна роботи полягає у впровадженні Lua-скриптів для створення гнучких та адаптивних правил виявлення загроз. Вперше запропоновано інтеграцію Lua у платформу Suricata для розширення функціональних можливостей, таких як виявлення порт-сканування з використанням нових алгоритмів і ведення детального логування HTTP-трафіку. Це дозволяє забезпечити високий рівень налаштування системи під специфічні вимоги користувача.

Практичне значення результатів роботи полягає у розробці рішення, яке дозволяє підвищити ефективність систем IDS/IPS у реальних умовах. Використання Lua дає змогу створювати адаптивні правила, інтегрувати додаткові функції в систему Suricata, а також забезпечити взаємодію із зовнішніми джерелами даних. Розроблене рішення було протестовано у реальному середовищі, що підтвердило його працездатність і ефективність у виявленні кіберзагроз, зокрема спроб порт-сканування.

Об'єктом дослідження є технології розширення функціональності систем виявлення та запобігання вторгнень. Предметом дослідження виступає використання мови Lua для створення користувацьких правил і алгоритмів аналізу у системі Suricata, а також для виявлення складних загроз і ведення логування подій у мережевому середовищі.

Апробація результатів магістерської роботи. Основні результати проведених досліджень обговорювались на: XII науково-технічна конференції «Інформаційні моделі, системи та технології».

Публікації. Основні результати кваліфікаційної роботи опубліковано у працях конференції (див. Додаток А).

РОЗДІЛ 1. АНАЛІЗ СУЧАСНОГО СТАНУ ТЕХНОЛОГІЙ IDS/IPS

1.1 Загальний огляд систем виявлення та запобігання вторгнень (IDS/IPS)

Системи виявлення та запобігання вторгнень (IDS/IPS) стали невід'ємною частиною сучасних мережевих інфраструктур, оскільки забезпечують критично важливу функцію захисту інформації. Їхній розвиток почався з появи перших комп'ютерних мереж, коли зловмисники почали активно використовувати вразливості для отримання несанкціонованого доступу до даних. Сучасний кіберпростір відзначається безпрецедентним рівнем складності та ризиків, що робить інтеграцію IDS/IPS важливою умовою безпеки будь-якої організації.

Типова архітектура систем IDS/IPS передбачає інтеграцію як мережевих (Network-based), так і хостових (Host-based) рішень для забезпечення багаторівневого захисту. Мережеві IDS/IPS розташовуються в ключових точках мережевої інфраструктури, наприклад, між маршрутизатором і фаєрволом, де вони аналізують увесь вхідний і вихідний трафік. Це дозволяє виявляти загрози на ранньому етапі та запобігати їх проникненню в мережу. Хостові IDS/IPS, у свою чергу, встановлюються безпосередньо на кінцевих пристроях, таких як сервери або робочі станції, і забезпечують локальний захист. Вони аналізують активність усередині операційної системи, такі як доступ до файлів чи процесів, що робить їх ефективними проти загроз, які можуть обійти мережевий захист. Комбінація мережевих і хостових рішень створює комплексну систему захисту, яка забезпечує як глобальний моніторинг мережі, так і детальний контроль на рівні окремих пристроїв [1].

На рисунку зображено типову архітектуру інтеграції систем IDS/IPS у мережеву інфраструктуру (див. рисунок 1.1). Основний елемент — Network-based IDS/IPS, розташований між маршрутизатором і фаєрволом, що дозволяє аналізувати трафік на рівні мережі. Також показано хостові IDS/IPS, які

працюють на окремих серверах і забезпечують локальний захист. Така архітектура дозволяє поєднувати моніторинг трафіку в масштабах мережі із захистом окремих хостів, створюючи багаторівневий підхід до безпеки.

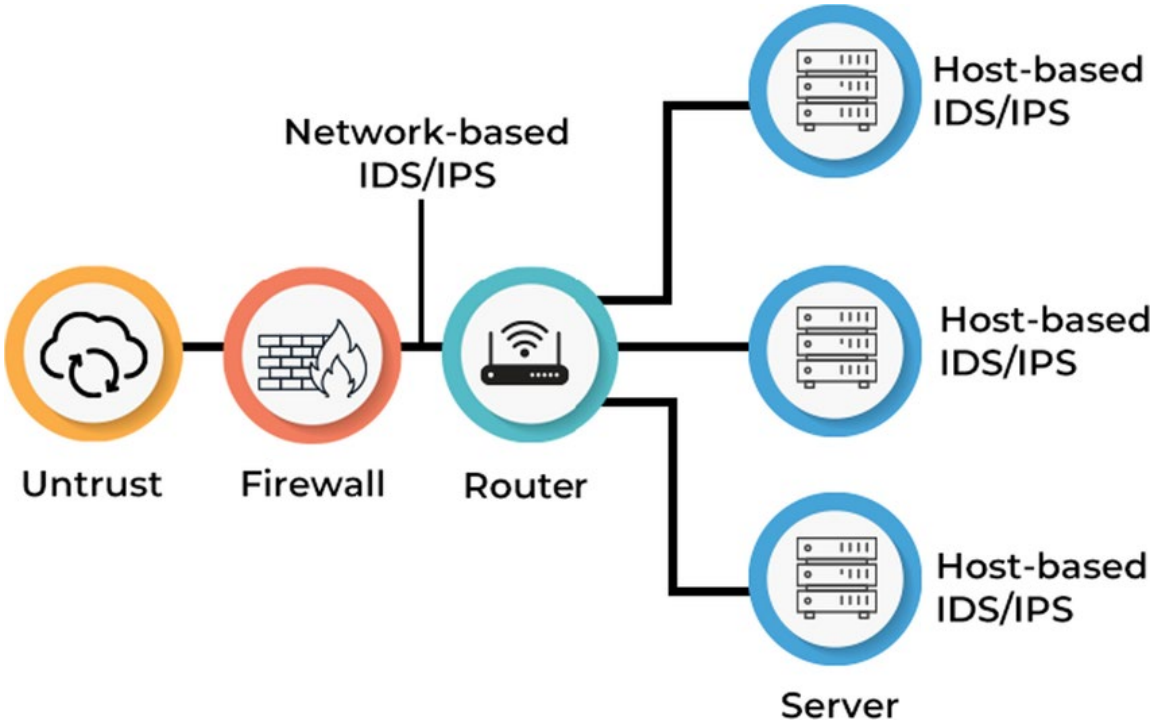

Рисунок 1.1 – Архітектура інтеграції систем IDS/IPS у мережеву інфраструктуру

Основною метою IDS є ідентифікація аномалій або підозрілих дій у мережевому трафіку. IDS виконує функцію моніторингу, збираючи та аналізуючи інформацію про події в мережі. Наприклад, коли система виявляє незвичайну активність, таку як повторювані спроби входу або підозрілий мережевий трафік, вона генерує оповіщення для адміністратора. IPS виконує схожі функції, але йде на крок далі, активно запобігаючи загрозам. Ця активна складова робить IPS більш ефективною у випадках, коли необхідно негайно зупинити атаку.

Існує декілька основних методів аналізу, які використовуються IDS/IPS. Перший – сигнатурний аналіз, при якому система порівнює поточний трафік із базою даних відомих атак. Цей метод забезпечує високий рівень точності

для виявлення відомих загроз, але має недоліки у випадках нових або модифікованих атак. Другий метод – аналіз на основі поведінки, який відстежує відхилення від нормальної роботи мережі. Цей підхід дозволяє виявляти нові типи атак, але може генерувати більше помилкових спрацьовувань.

Ці два підходи можуть працювати незалежно або спільно, що дозволяє поєднувати точність виявлення відомих атак із можливістю виявлення нових, невідомих загроз. Рисунок 1.2 ілюструє принцип роботи двох основних підходів виявлення загроз, які використовуються в системах IDS/IPS: сигнатурного (Signature-based Detector) і поведінкового або аномального (Anomaly-based Detector) аналізу (див. рисунок 1.2).

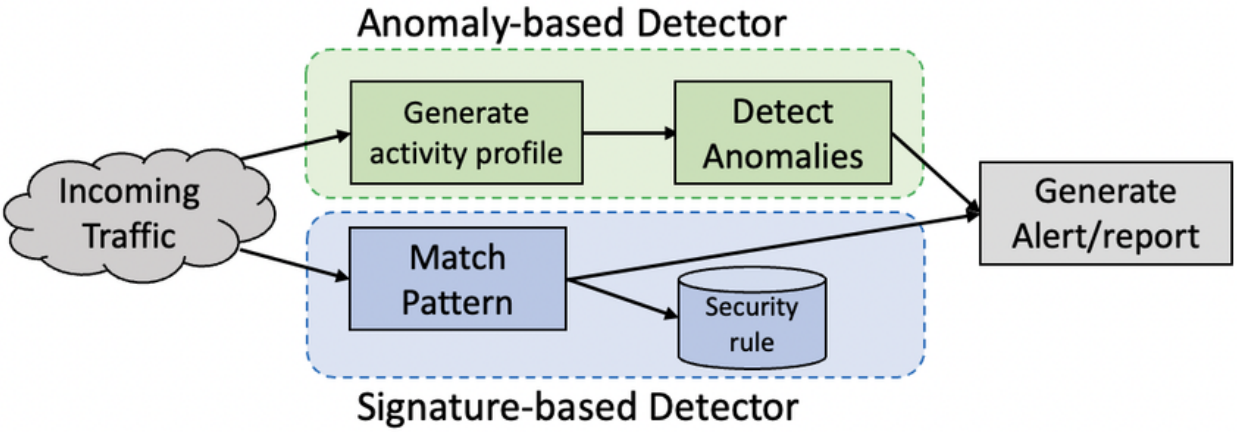

Рисунок 1.2 – Принцип роботи сигнатурного і аномального аналізу

Завдяки широкому розповсюдженню мережевих технологій та цифровізації бізнес-процесів, зловмисники мають все більше можливостей для здійснення атак. Наприклад, атаки типу "людина посередині" (Man-in-the-Middle), відмови в обслуговуванні (DDoS) або шкідливе програмне забезпечення становлять загрозу як для бізнесу, так і для приватних користувачів. Системи IDS/IPS дозволяють виявляти та запобігати таким загрозам у реальному часі, що значно підвищує загальний рівень кібербезпеки.

Окремої уваги заслуговує питання інтеграції сучасних технологій у системи IDS/IPS. У сучасних умовах ефективність таких систем значною мірою залежить від можливості їхньої адаптації до нових загроз. Наприклад, використання мови Lua дозволяє розширити функціональні можливості IDS/IPS, додати користувацькі правила виявлення та реалізувати складні алгоритми аналізу даних. Ця мова сценаріїв стала популярною завдяки своїй легкості, простоті та гнучкості [2].

Інші переваги використання Lua в IDS/IPS включають можливість створення специфічних правил для аналізу окремих типів протоколів, інтеграцію з іншими системами кібербезпеки, а також налаштування системи під потреби конкретної організації. Наприклад, у випадку використання платформи Suricata, Lua-скрипти можуть бути інтегровані для створення користувацьких правил, які не лише виявляють підозрілу активність, але й дозволяють детально аналізувати специфічні протоколи або види трафіку.

Сьогодні IDS/IPS є не просто інструментами виявлення загроз, а основою для створення комплексних систем управління безпекою. Їхня інтеграція з іншими компонентами кібербезпеки, такими як SIEM-системи, дозволяє створити більш повний та ефективний підхід до захисту інформації. IDS/IPS стають основним джерелом даних для аналізу подій у мережі, що допомагає швидко реагувати на інциденти та мінімізувати їхні наслідки.

Іншим важливим напрямком розвитку сучасних IDS/IPS є пристосування до нових парадигм мережевої взаємодії. Зокрема, стрімке впровадження віртуалізації, контейнеризації та хмарних сервісів призводить до появи динамічних, розподілених та масштабованих інфраструктур. В умовах, коли мережеві топології стають дедалі складнішими, а обчислювальні ресурси та сервіси можуть постійно мігрувати між фізичними та віртуальними середовищами, IDS/IPS повинні забезпечувати:

– гнучку інтеграцію з віртуальними комутаторами, контейнерними оркестраторами (наприклад, Kubernetes) та гібридними хмарними платформами

– можливість централізованого керування політиками безпеки, яке синхронізується з динамічними змінами в інфраструктурі

– підтримку розширеного моніторингу у середовищах з високою щільністю сервісів, де трафік може бути зашифрованим та проходити через комплексні ланцюги сервіс-мереж (service mesh)

Важливою тенденцією є також поява ширшого спектра протоколів і сервісів, які потребують специфічного аналізу. IDS/IPS системи поступово вчаться не лише обробляти класичний HTTP-трафік, але й ефективно протидіяти загрозам у середовищах інтернету речей (IoT), промислових мереж (ICS/SCADA), 5G/SDN та інших спеціалізованих інфраструктурах. Це може передбачати адаптацію правил, оновлення сигнатур, застосування поведінкових аналітик та використання інтегрованих скриптових сценаріїв для врахування специфіки протоколів, які раніше були нетиповими для класичних мереж [3].

Окремою проблематикою є аналіз зашифрованого трафіку (HTTPS, TLS 1.3 та інші протоколи шифрування). Зростання частки зашифрованого трафіку суттєво ускладнює роботу IDS/IPS, оскільки традиційні методи глибокої інспекції пакетів (DPI) стають малоефективними. Відповідно, сучасні IDS/IPS:

– використовують методи аналізу метаданих (наприклад, розгляд розмірів пакетів, частоти звернень, часових проміжків) для виявлення аномалій у зашифрованому середовищі

– інтегруються з SSL/TLS термінацією або застосовують технології розподіленого дешифрування на спеціалізованих пристроях чи сервісах

– поєднують поведінковий аналіз з інформацією про репутацію доменів та IP-адрес, отриманою з зовнішніх джерел

Додатково заслуговують уваги питання відповідності IDS/IPS регуляторним вимогам та стандартам безпеки. Для низки галузей (фінансовий сектор, охорона здоров'я, енергетика) вимоги до обробки, зберігання та передачі даних можуть бути особливо суворими. Це означає, що IDS/IPS повинні:

– надавати детальні журнали подій та архівувати їх для аудиту відповідно до норм і стандартів (наприклад, PCI DSS, GDPR, HIPAA)

– підтримувати механізми звітності та доступу до історії інцидентів, інтегруючись з SIEM та іншими засобами комплексного моніторингу інфраструктури

– забезпечувати можливість швидко реагувати на виявлені порушення, не лише блокуючи трафік, а й допомагаючи командам безпеки аналізувати корінні причини інцидентів

Наведені вище аспекти підкреслюють, що сучасні IDS/IPS системи перестають бути статичними інструментами, сконцентрованими виключно на фіксованих правилах. Вони перетворюються на динамічні, розумні та контекстно обізнані платформи, здатні ефективно взаємодіяти з іншими компонентами безпеки, адаптуватися до мінливих умов та забезпечувати багаторівневий підхід до протидії загрозам [4].

Нижче подано таблицю 1.1, яка узагальнює основні чинники, що впливають на розвиток та підвищення ефективності IDS/IPS (див. таблицю 1.1):

Таблиця 1.1 – Чинники розвитку сучасних IDS/IPS систем

Чинник	Результат для IDS/IPS
Динамічні мережеві інфраструктури (віртуалізація, хмари, контейнеризація)	Гнучка інтеграція з віртуальними комутаторами та оркестраторами, можливість централізованого керування політиками безпеки
Зростання обсягу та шифрування трафіку	Необхідність аналізу метаданих, інтеграції TLS/SSL термінації, поведінковий підхід до виявлення аномалій
Спеціалізовані та нетипові протоколи (IoT, ICS/SCADA, 5G)	Адаптація правил, застосування скриптових мов для обробки нестандартного трафіку, розширення функціоналу

Продовження таблиці 1.1

Чинник	Результат для IDS/IPS
Підвищення регуляторних вимог та стандартів безпеки	Налагоджена звітність, аудит, деталізація журналів подій, інтеграція з SIEM для відповідності нормам
Інтеграція з платформами управління безпекою	Об’єднання даних з IDS/IPS з інформацією про загрози, кореляція подій, автоматизація реагування за допомогою SOAR

1.2 Переваги та недоліки використання скриптових мов у IDS/IPS

Скриптові мови відіграють важливу роль у розширенні функціональних можливостей сучасних систем IDS/IPS. Їх використання забезпечує більшу гнучкість, адаптивність і ефективність при вирішенні завдань виявлення та запобігання загрозам. Однією з найбільш популярних скриптових мов, яка інтегрується в IDS/IPS, є Lua. Ця мова вирізняється простотою у використанні, невеликим обсягом, високою продуктивністю та легкістю інтеграції в інші системи.

Переваги використання скриптових мов у IDS/IPS можна поділити на кілька ключових категорій. По-перше, вони дозволяють створювати користувацькі правила та алгоритми для виявлення специфічних загроз. Наприклад, за допомогою Lua можна створити сценарій, який аналізує HTTP-трафік і виявляє запити до небезпечних ресурсів, таких як PHP-файли, що часто стають об'єктом атак. Такі правила значно підвищують ефективність роботи IDS/IPS у специфічних сценаріях використання. По-друге, скриптові мови надають можливість динамічно змінювати налаштування системи. Це особливо важливо в умовах, коли зловмисники постійно змінюють свої тактики, методи та засоби атаки. У таких випадках використання скриптових мов дозволяє оперативно вносити зміни до правил виявлення без необхідності оновлювати основний код системи [5].

Ще однією значною перевагою є можливість обробки нестандартних протоколів або трафіку. У стандартних IDS/IPS існують недоліки щодо

підтримки деяких рідкісних або спеціалізованих протоколів. Скриптові мови, такі як Lua, дозволяють реалізовувати власну логіку обробки, що забезпечує більшу універсальність системи. Це важливо для організацій, які використовують специфічні мережеві протоколи або унікальні типи трафіку.

Окрім цього, інтеграція скриптових мов підвищує масштабованість систем IDS/IPS. Наприклад, у великих мережах, де генерується значний обсяг трафіку, правила на основі сценаріїв дозволяють більш ефективно фільтрувати дані та зменшувати навантаження на систему. За рахунок цього можна досягти більш високої продуктивності навіть у складних мережевих середовищах.

Однак, попри численні переваги, використання скриптових мов у IDS/IPS має і свої недоліки. Одним із них є потенційна складність у створенні та налаштуванні сценаріїв. Для того, щоб написати ефективний сценарій, розробник повинен мати ґрунтовні знання про скриптові мови, мережеві протоколи та типи атак. Це може створити труднощі для менш досвідчених фахівців з інформаційної безпеки [6] .

Іншими недоліками є можливий вплив на продуктивність системи. Складні сценарії, які виконують глибокий аналіз трафіку або використовують ресурсоємні алгоритми, можуть значно знижувати швидкість обробки даних. Це може бути критичним у високонавантажених мережах, де IDS/IPS повинна працювати в режимі реального часу.

Також слід враховувати ризики, пов’язані з безпекою самих сценаріїв. Помилки або вразливості у написаних скриптах можуть бути використані зловмисниками для обходу захисту або навіть для атак на систему IDS/IPS. Тому важливо забезпечити ретельне тестування та перевірку кожного сценарію перед його впровадженням у реальне середовище.

Таким чином, використання скриптових мов у IDS/IPS є потужним інструментом для підвищення ефективності та гнучкості цих систем. Однак для досягнення максимальної ефективності важливо враховувати як переваги,

так і недоліки, які є переліченими у таблиці 1.2, забезпечуючи баланс між функціональністю, продуктивністю та безпекою (див. таблицю 1.2).

Таблиця 1.2 – Порівняння переваг та недоліків використання скриптових мов у IDS/IPS

Переваги	Недоліки
Можливість створення користувацьких правил для специфічних загроз.	Високий рівень знань потрібний для розробки сценаріїв.
Можливість оперативного внесення змін у правила без перезавантаження системи.	Зміни в скриптах можуть призвести до помилок у роботі системи.
Дозволяє обробляти специфічні типи трафіку, які не підтримуються стандартними правилами.	Складність у реалізації логіки для рідкісних або складних протоколів.
Зменшення навантаження на систему за рахунок локальної обробки трафіку скриптами.	Використання ресурсомістких алгоритмів може знижувати продуктивність у високонавантажених мережах.
Додатковий рівень контролю за специфічними загрозами.	Уразливості в скриптах можуть бути використані зловмисниками.

Додатковим аспектом, який заслуговує на увагу, є поєднання скриптових мов з іншими технологіями, що сприяють підвищенню ефективності IDS/IPS. Наприклад, використання інструментів оркестрації та автоматизації (SOAR-платформи) дозволяє застосовувати створені на основі скриптових мов правила та логіку в розподілених середовищах. Це особливо актуально для великих організацій, де сотні або тисячі сенсорів IDS/IPS повинні підтримувати єдиний набір динамічних політик безпеки [7].

Іншим важливим напрямком є інтеграція скриптових мов з методами машинного навчання та штучного інтелекту. Створені вручну скрипти можуть працювати в тандемі з алгоритмами ML/AI, використовуючи їхні результати для уточнення логіки прийняття рішень. Наприклад, якщо ML-модуль оцінює певний трафік як потенційно шкідливий, скриптова логіка може перевірити додаткові ознаки або звернутися до зовнішніх джерел розвідданих, перш ніж

заблокувати з’єднання. Таким чином, поєднання скриптів і інтелектуальних моделей аналізу трафіку дає змогу автоматично адаптувати IDS/IPS до нових типів загроз [8].

Важливо також зазначити, що наявність гнучкого скриптового шару відкриває двері до реалізації "User-Defined Functions" (UDF) та розширень, орієнтованих на галузеві стандарти. Наприклад, у деяких секторах, таких як промислові мережі (ICS/SCADA) або інтернет речей (IoT), особливо цінним є створення вузькоспеціалізованих правил, які враховують протоколи чи формати даних, невідомі загальним IDS/IPS. За допомогою скриптових мов можна швидко додати підтримку нових типів пакетів, специфічних команд контролерів або датчиків, які виходять за межі традиційних мережевих служб.

Доцільним є також забезпечення розвинених механізмів безпеки та контролю якості для скриптів. Розробники та адміністратори IDS/IPS можуть впроваджувати концепції "code review" та "linting" для перевірки скриптів перед їх впровадженням у продуктивне середовище. Можливе застосування спеціальних тестових стендів, де кожен новий сценарій піддається імітації атаки, аналізу стабільності та вимірюванню впливу на продуктивність. Це забезпечує більш безпечне та передбачуване впровадження змін [9].

Нижче наведено таблицю 1.3, яка ілюструє потенційні напрямки розширення функціональності IDS/IPS за допомогою скриптових мов у майбутньому (див. таблицю 1.3):

Таблиця 1.3 – Потенційні напрями використання скриптових мов для подальшого розвитку IDS/IPS

Напрямок	Опис	Приклад реалізації
Інтеграція з ML/AI	Поєднання скриптових правил із результатами машинного навчання та інтелектуального аналізу загроз	Використання Lua-скрипта, що реагує на "оцінку ризику" від ML-модуля

Продовження таблиці 1.3

Напрямок	Опис	Приклад реалізації
Розширення галузевої специфіки	Створення унікальних правил для специфічних протоколів, що використовуються у промисловості чи IoT	Написання власних аналізаторів трафіку SCADA-протоколів
Оркестрація та автоматизація	Масштабування логіки безпеки через SOAR-платформи та централізоване керування сценаріями	Використання Ansible чи Terraform для розгортання однакових скриптів на багатьох сенсорах
Тестування та валідація	Створення середовищ для попереднього тестування та перевірки якості скриптів перед інтеграцією	Автоматизований "CI/CD pipeline" для сценаріїв з тестами та статичним аналізом коду
Розвиток екосистеми доповнень	Спільноти розробників діляться готовими сценаріями та плагінами, що розширюють функціонал IDS/IPS	Публікація користувацьких сценаріїв у спільних репозиторіях (GitHub, GitLab)

1.3 Використання машинного навчання та штучного інтелекту для підвищення ефективності IDS/IPS

Використання машинного навчання та штучного інтелекту для підвищення ефективності IDS/IPS є одним із ключових напрямків розвитку сучасних систем кібербезпеки. Класичні підходи до виявлення загроз, які зосереджуються переважно на сигнатурному аналізі або простому поведінковому відстеженні аномалій, дедалі частіше виявляються недостатніми для успішного протистояння новим, більш витонченим методам атак. Машинне навчання, засноване на статистичних методах та сучасних архітектурах штучного інтелекту, здатне значно розширити можливості IDS/IPS, підвищуючи їхню точність, гнучкість та адаптивність [10].

Основна ідея застосування ML/AI у IDS/IPS полягає у можливості навчати системи розпізнавати як відомі, так і невідомі типи атак, уникаючи повної залежності від статичних сигнатур. За допомогою алгоритмів класифікації, кластеризації або виявлення аномалій стає можливою обробка величезних обсягів даних про трафік, сесії, події безпеки та контекстуальну інформацію. Застосування ML дозволяє системам динамічно адаптуватися до

нових загроз, автоматично оновлювати моделі виявлення атак, враховувати складні кореляції та мінімізувати кількість помилкових спрацьовувань.

Одним із головних викликів застосування ML/AI для IDS/IPS є коректна підготовка навчальних вибірок та постійне оновлення даних. Система, що ґрунтується на штучному інтелекті, вимагає регулярного перенавчання та уточнення моделей, щоб не втратити актуальність і не стати вразливою до нових методів обходу захисту. Важливим аспектом є використання релевантних ознак трафіку та подій. Наприклад, для аналізу може братися не лише вміст пакетів, але й метадані з'єднань, тимчасові закономірності, інформація про джерело та призначення, історія попередніх взаємодій між хостами, результати попередньої обробки за допомогою сценаріїв Lua тощо [11].

Використання машинного навчання у IDS/IPS може бути реалізоване різними способами:

– застосування класичних методів класифікації, таких як логістична регресія, дерева рішень, випадковий ліс чи метод опорних векторів, для виявлення відомих патернів атак

– впровадження неконтрольованих методів, таких як алгоритми кластеризації або дерев аномалій, для виявлення невідомих загроз, що не відповідають жодній сигнатурі

– використання глибоких нейронних мереж та архітектур глибинного навчання для аналізу складних патернів у великих потоках мережевого трафіку, враховуючи контекст та історичні дані про інциденти

– інтеграція зовнішніх джерел розвідданих та баз знань, що дозволяє системам ML/AI отримувати актуальну інформацію про нові експлойти, ботнети, фішингові домени, пов'язані з ними атрибути та поведінкові характеристики

– використання гібридних підходів, коли ML-моделі поєднуються з традиційними сигнатурними та поведінковими методами, а також з гнучкими сценаріями на Lua для досягнення максимальної точності та продуктивності

Одним із перспективних напрямків є поєднання ML/AI можливостей із розширюваністю, яку надають скриптові мови, зокрема Lua. Використання Lua в IDS/IPS дозволяє реалізувати динамічні правила, що запускаються на підставі результатів аналізу машинного навчання. Наприклад, модуль ML може здійснювати попередню обробку та класифікацію трафіку, а Lua-скрипт – приймати рішення про те, як реагувати на виявлену активність. Це відкриває можливості створення гнучких сценаріїв, які враховують не лише статичні сигнатури або порогові значення, але й результати інтелектуального аналізу. Таким чином, Lua може виступати інструментом "логіки прийняття рішень", додаючи до ML-модулів адаптивну компоненту, що дозволяє оперативно змінювати політику захисту залежно від нових даних та типів атак [12].

Важливим аспектом взаємодії між ML/AI та Lua у IDS/IPS є створення інтерфейсів взаємодії, які дозволять ML-модулям передавати результати свого аналізу Lua-скриптам. Наприклад, ML-модель, побудована на основі глибоких нейронних мереж, може за певний проміжок часу формувати список підозрілих сесій та характеристик підозрілого трафіку. Далі Lua-скрипт використовує цю інформацію для прийняття оперативних рішень щодо блокування, сповіщення адміністратора або додаткового аналізу. Такий підхід значно підвищує рівень адаптації системи, адже реагування відбувається не лише на заздалегідь відомі патерни, а й на абсолютно нові ознаки та сигнали.

Ще одним чинником, що робить ML/AI актуальними для IDS/IPS, є необхідність роботи у високонавантажених середовищах. Обробка великих масивів трафіку та подій у реальному часі вимагає високої продуктивності. Сучасні ML-алгоритми, налаштовані на специфіку мережевого середовища, можуть бути оптимізовані для обробки потоків даних у режимі "on-the-fly", використовуючи розподілені обчислювальні ресурси та апаратне прискорення

(наприклад, GPU або FPGA). Це дозволяє IDS/IPS не лише виявляти загрози точніше, але й працювати зі значно більшими обсягами даних, не втрачаючи продуктивності.

Цікаво, що інтеграція ML/AI у IDS/IPS дає змогу розробляти метрики оцінки роботи системи на основі адаптивних критеріїв. Замість того, щоб спиратися лише на класичні показники "false positives" та "false negatives", можна оцінювати ефективність системи за допомогою складніших метрик, що враховують динамічну природу загроз. Наприклад, ML-моделі можуть надавати оцінки рівнів ризику, а Lua-скрипти – сценарно реагувати на зміни цих рівнів, тим самим коригуючи політику безпеки.

Таблиця 1.4 – Порівняння класичного та ML/AI підходів у IDS/IPS

Критерій	Класичний підхід	ML/AI підхід
Орієнтація	На відомі сигнатури та патерни	На виявлення нових та невідомих загроз
Адаптивність	Обмежена, залежить від ручних оновлень	Висока, базується на автоматичному навчанні моделей
Чутливість до нових атак	Низька	Висока, алгоритми можуть виявляти ознаки невідомих атак
Кількість помилкових спрацювань	Залежить від якості сигнатур	Може бути оптимізована алгоритмами навчання
Масштабованість	Обмежена, важко розширювати під нові протоколи	Висока, моделі можуть обробляти різні протоколи та середовища
Інтеграція з Lua	Можлива, але обмежена логікою сигнатур	Органічна, ML моделі надають результати, які Lua-скрипти використовують для динамічних рішень

Об’єднуючи ML/AI підходи з можливостями розширення, що їх надає Lua, IDS/IPS системи стають більш гнучкими, інтелектуальними та здатними відповідати на постійно мінливі умови кіберпростору. Якщо класичні сигнатурні методи захисту нагадують статичні "пастки", розставлені по периметру, то ML/AI, разом із Lua-сценаріями, перетворюють IDS/IPS на "розумних агентів", здатних навчатися, вчитися на помилках, реагувати на нові виклики та автоматично удосконалювати власні методи виявлення. Це відкриває перспективи створення більш надійних, продуктивних та адаптивних систем захисту, які відповідають на сучасні загрози з максимальним рівнем ефективності.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ ТЕОРЕТИЧНИХ ОСНОВ РОЗШИРЕННЯ IDS/IPS ЗА ДОПОМОГОЮ LUA

2.1 Основи мови Lua: переваги, можливості інтеграції та особливості використання

Мова програмування Lua — це компактна, швидка та гнучка скриптова мова, розроблена спеціально для вбудовування в інші програми. Вона була створена на початку 1990-х років у Федеральному університеті Ріо-де-Жанейро (Бразилія) для забезпечення легкого і гнучкого способу розширення можливостей існуючого програмного забезпечення. Її особливості зробили Lua ідеальним вибором для інтеграції у різні програмні платформи, зокрема системи IDS/IPS, такі як Suricata [13].

Lua відрізняється від інших скриптових мов своєю мінімалістичністю та високою продуктивністю. Вона має компактний інтерпретатор і простий синтаксис, який легко освоюється навіть новачками. Lua працює на всіх популярних платформах, включаючи Windows, macOS, Linux, а також на вбудованих системах, що робить її універсальним інструментом для розробки та інтеграції.

Сучасні IDS/IPS системи невпинно розвиваються під впливом нових типів загроз, технологій та парадигм кіберзахисту. Ці інструменти мають забезпечити не лише виявлення аномалій у трафіку, а й гнучке реагування на постійно змінювані умови. Оскільки класичні сигнатурні та поведінкові методи аналізу не завжди достатні для повноцінного виявлення складних та багатоступеневих атак, розробники шукають нові шляхи розширення функціональності IDS/IPS. Одним із найефективніших інструментів у цьому напрямку стала інтеграція мови Lua.

Lua використовується у різних IDS/IPS системах, таких як Suricata, через її легкість, продуктивність і простоту інтеграції. Аналітичні дослідження в області інформаційної безпеки показують, що використання Lua дозволяє

значно підвищити ефективність виявлення загроз за рахунок створення специфічних правил аналізу трафіку. Наприклад, у Suricata Lua-скрипти можуть виконувати завдання, які стандартні правила не можуть реалізувати, такі як обробка нестандартних протоколів, інтеграція з базами даних загроз або виконання складних поведінкових аналізів.

Основні переваги Lua:

–легкість та компактність: Lua відома своєю невеликою «вагою» та мінімальним використанням ресурсів. Це важливо для IDS/IPS, які працюють у режимі реального часу, часто під значним навантаженням. Вбудовування Lua-інтерпретатора в ядро системи зазвичай не потребує суттєвого збільшення апаратних ресурсів.

–висока продуктивність: архітектура мови Lua та її інтерпретатор оптимізовані для швидкої обробки скриптів. Для IDS/IPS це має критичне значення, адже будь-яке затримання аналізу трафіку може призвести до пропуску загроз або зниження пропускної здатності мережі.

–гнучкість у створені спеціалізованих правил: мова Lua дозволяє розробникам писати власні логічні конструкції, які виходять за рамки статичних сигнатур. Наприклад, можна створити скрипт, що аналізує послідовність мережевих сесій між двома хостами та виявляє підозрілі патерни, які не фігурують у стандартних базах сигнатур. Це дає змогу виявляти атаки «нульового дня» або складні багатокрокові інциденти.

–модульність та розширюваність: Lua легко інтегрується з іншими компонентами IDS/IPS та стороннімі сервісами. За потреби можна звертатися до зовнішніх баз даних загроз (Threat Intelligence feeds), виконувати складні перевірки достовірності даних чи навіть взаємодіяти з зовнішніми API. Таким чином IDS/IPS перестає бути закритою системою, а перетворюється на гнучку платформу, орієнтовану на динамічну протидію загрозам.

Основною перевагою Lua є її легкість. Інтерпретатор Lua займає всього кілька сотень кілобайт, що дозволяє легко інтегрувати його у вже існуючі

системи без значного впливу на їхню продуктивність. Lua забезпечує високий рівень продуктивності завдяки використанню ефективного інтерпретатора байт-коду та оптимізованого компілятора. Це дозволяє виконувати складні сценарії в реальному часі, що є критично важливим для систем IDS/IPS, які повинні реагувати на загрози з мінімальними затримками [14].

Lua підтримує модульну архітектуру, що дозволяє розширювати її можливості за допомогою додаткових бібліотек. Наприклад, у контексті Suricata Lua може використовуватися для створення специфічних правил виявлення загроз, аналізу нестандартних протоколів або інтеграції з іншими системами кібербезпеки. Мова також забезпечує підтримку інтерактивного налагодження, що значно спрощує процес розробки та тестування скриптів.

Особливості Lua включають підтримку таблиць як базової структури даних, легку роботу з функціями та простий механізм управління пам'яттю за допомогою автоматичного збирання сміття (garbage collection). Це дозволяє створювати ефективні та масштабовані сценарії, які можуть працювати з великими обсягами даних без значних витрат ресурсів.

Іншою важливою функцією Lua є підтримка інтеграції з базами даних загроз. Наприклад, Lua-скрипти можуть звертатися до зовнішніх API для отримання актуальної інформації про відомі вразливості або потенційні загрози. Це забезпечує можливість швидкої адаптації системи до нових атак без необхідності внесення змін до основного коду Suricata [15].

Приклади використання Lua у IDS/IPS:

–обробка нестандартних або власних протоколів: часто організації використовують нетипові або внутрішньокорпоративні протоколи, що не відображені у стандартних наборах сигнатур. За допомогою Lua можна створити спеціальний модуль аналізу такого трафіку, наприклад, розпізнавати нестандартні заголовки пакетів, структуру даних або поведінкові особливості. Це дозволяє IDS/IPS адаптуватися до середовища конкретної організації.

–поведінковий аналіз та кореляція подій: Lua-скрипти можуть

застосовуватися для реалізації складних логік, наприклад, аналізу послідовності HTTP-запитів до веб-сервера. Можна перевіряти, чи відбувається підозріле звертання до адмін-панелі, нестандартних файлів (наприклад, shell.php або adminer.php) чи спроба SQL-ін’єкцій. Замість статичного порівняння зі списком сигнатур Lua дає змогу виявити закономірності, які свідчать про підготовку до атаки або сканування вразливостей.

– інтеграція із системами SIEM, Threat Intelligence та SOAR: написані на Lua скрипти можуть динамічно звертатися до зовнішніх джерел розвідданих (наприклад, API сервісів, які надають інформацію про IP-адреси, пов'язані з ботнетами чи фішинговими доменами) та приймати рішення залежно від актуального контексту. Якщо IDS/IPS виявляє підозрілий трафік, скрипт на Lua може запросити у TI-платформи інформацію про цей домен або IP. При виявленні негативної репутації – підняти рівень критичності або негайно заблокувати з'єднання. Інтеграція з SIEM дає можливість передавати результати аналізу для подальшої кореляції з іншими подіями безпеки.

– оптимізація та розвантаження основних правил: Lua-скрипти можуть виконувати первинну фільтрацію потоку даних, відсіваючи очевидно нешкідливий трафік та передаючи до основних модулів IDS/IPS лише потенційно небезпечні пакети. Такий підхід знижує навантаження на основні движки аналізу, підвищуючи загальну продуктивність і масштабованість системи. Це особливо корисно у великих мережах з високим обсягом трафіку.

Одним із ключових застосувань Lua у Suricata є розширення функціональності системи шляхом створення складних правил виявлення загроз. Наприклад, Lua дозволяє аналізувати HTTP-запити, визначати параметри, що використовуються для виконання атак типу SQL-ін'єкцій, або моніторити поведінкові патерни у трафіку. Lua також може бути використана для реалізації спеціальних алгоритмів, таких як перевірка відповідності вхідного трафіку базі IP-адрес з низькою репутацією (див. таблицю 2.1).

Таблиця 2.1 – Приклади основних можливостей використання Lua

Можливість	Опис	Приклад використання
Створення користувацьких правил	Lua дозволяє визначати складні умови для виявлення специфічних загроз.	Написання скрипта, що аналізує HTTP-запити для виявлення шкідливих параметрів.
Інтеграція з SIEM-системами	Можливість формування спеціалізованих логів для подальшого аналізу в SIEM.	Створення Lua-скрипта, який генерує логи у форматі, сумісному з конкретною SIEM-системою.
Обробка специфічних протоколів	Lua дозволяє розширювати підтримку нестандартних або власних протоколів.	Розробка скрипта для аналізу власного мережевого протоколу, не підтримуваного стандартно в Suricata.

Скриптові мови стали важливим інструментом для розширення можливостей IDS/IPS, забезпечуючи гнучкість, адаптивність і можливість створення специфічних правил для виявлення загроз. Вибір конкретної мови для інтеграції в платформу залежить від багатьох факторів, таких як продуктивність, легкість інтеграції, обсяг використовуваних ресурсів і складність у використанні. Найбільш поширеними мовами для таких завдань є Lua, Python та Perl.

Lua є однією з найоптимальніших мов для інтеграції у системи IDS/IPS, такі як Suricata. Її основні переваги включають високу продуктивність, компактний інтерпретатор і простий синтаксис, який легко освоїти навіть новачкам. Крім того, Lua займає мінімум пам'яті, що робить її ідеальною для роботи в реальному часі, де важлива швидкість реакції на загрози. Завдяки своїй легкості, Lua забезпечує безпроблемну інтеграцію з Suricata, надаючи розробникам можливість створювати специфічні правила для аналізу трафіку або обробки нестандартних протоколів.

Python, з іншого боку, надає розширений функціонал і можливості для роботи з великими обсягами даних, але за рахунок зниженої продуктивності. Хоча Python використовується в багатьох рішеннях для кібербезпеки, його застосування в IDS/IPS може бути обмеженим через значний обсяг інтерпретатора і більш високі вимоги до ресурсів. Python добре підходить для

аналітики або обробки логів, але для задач реального часу він менш ефективний, ніж Lua.

Perl, яка була однією з перших мов сценаріїв, що широко використовувалися в мережевій безпеці, також має свої переваги, зокрема багатий набір інструментів для обробки тексту. Проте через складність синтаксису та знижену продуктивність порівняно з Lua, її популярність у сучасних IDS/IPS зменшилася. Perl вимагає більш складної адаптації під конкретні платформи, що може створювати додаткові труднощі при її використанні.

У таблиці 2.2 наведено порівняння трьох популярних скриптових мов для використання в IDS/IPS, зокрема в контексті їхньої інтеграції та продуктивності (див. таблицю 2.2):

Таблиця 2.2 – Порівняння інтеграції Lua з іншими скриптовими мовами у контексті IDS/IPS

Характеристика	Lua	Python	Perl
Продуктивність	Висока завдяки оптимізованому інтерпретатору.	Середня, залежить від середовища виконання.	Середня, знижена через більший обсяг виконуваного коду.
Обсяг інтерпретатора	~200 КБ	~30 МБ	~10 МБ
Легкість інтеграції	Легка інтеграція в платформи, як-от Suricata.	Часто вимагає додаткових модулів.	Вимагає глибшої адаптації під конкретні платформи.
Складність навчання	Проста для початківців, мінімалістичний синтаксис.	Більш складна для новачків.	Середня, багато функцій для роботи з текстом.
Використання пам'яті	Ефективне	Більш ресурсоємне	Помірне

2.2 Огляд механізмів розширення Suricata за допомогою Lua-скриптів

Suricata є однією з найбільш потужних і гнучких платформ для виявлення та запобігання вторгнень (IDS/IPS), а також для моніторингу мережевого трафіку. Ця система була розроблена Open Information Security Foundation (OISF) як відкрите програмне забезпечення, що забезпечує багатофункціональний аналіз мережевого трафіку, високу продуктивність і розширюваність за допомогою скриптових мов, таких як Lua.

Suricata інтегрується у мережеву інфраструктуру, забезпечуючи моніторинг трафіку між зовнішнім інтернетом та внутрішньою мережею. Вона виконує функції аналізу пакетів, ідентифікації загроз і формування сповіщень для адміністратора безпеки. На рисунку 2.1 зображена типова схема розгортання Suricata в контексті мережевої безпеки (див. рисунок 2.1). Suricata розташовується між фаєрволом та внутрішньою мережею, де вона виконує функції моніторингу і аналізу трафіку. Вхідний трафік з інтернету через фаєрвол спрямовується на Suricata, де аналізується на предмет потенційних загроз. Якщо трафік відповідає правилам безпеки, він передається до веб-сервера (наприклад, із внутрішньою IP-адресою 192.168.1.2/24). У разі виявлення підозрілої активності або загроз Suricata генерує сповіщення (alerts), які передаються адміністратору або в SIEM-систему для подальшого аналізу.

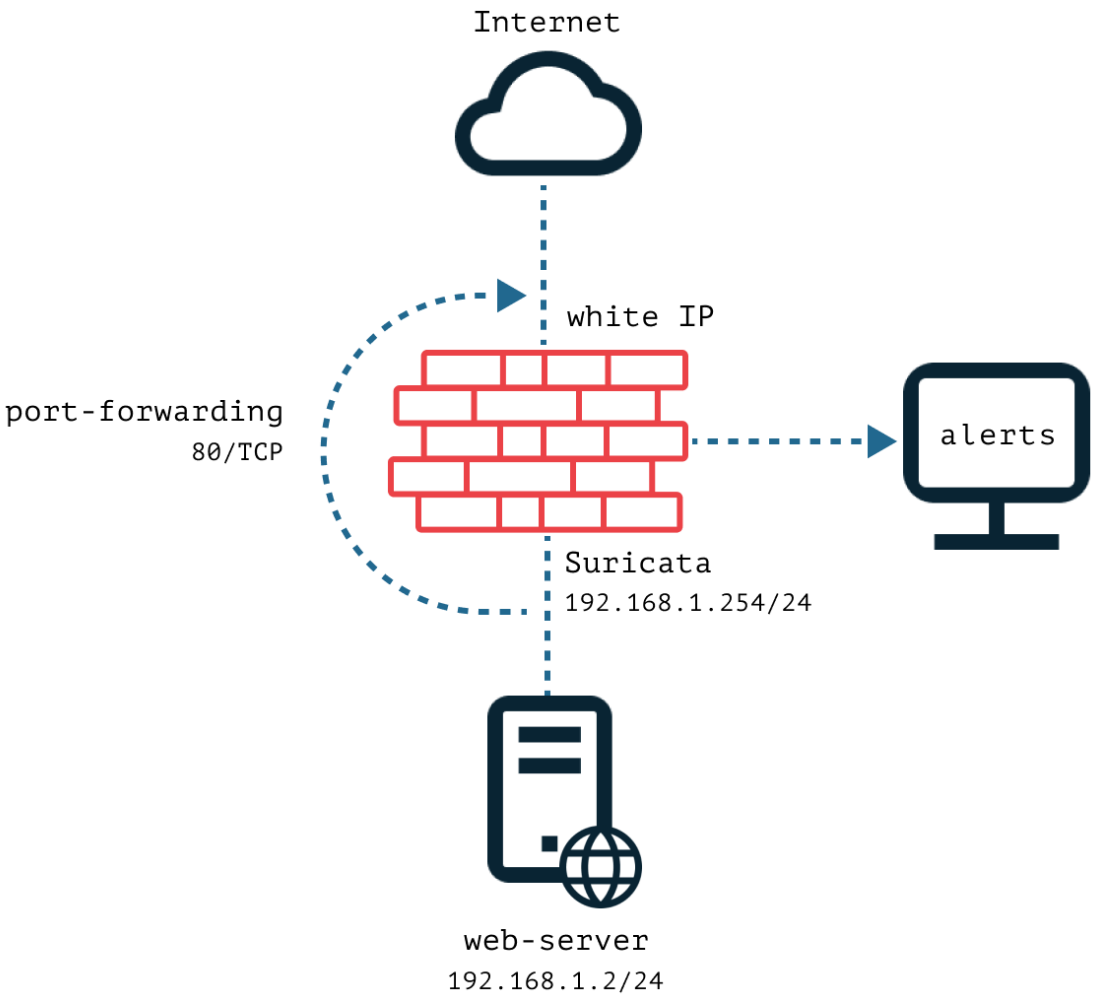

Рисунок 2.1 – Базова архітектура інтеграції Suricata у мережеву інфраструктуру

Suricata відзначається багатопотоковою архітектурою, яка дозволяє системі максимально використовувати апаратні ресурси багатоядерних процесорів. Це досягається шляхом розподілу обробки мережевого трафіку між кількома потоками, які виконують завдання паралельно. Наприклад, один потік може бути відповідальним за захоплення трафіку, інший — за його розпакування, а третій — за застосування правил виявлення. Такий підхід дозволяє Suricata працювати з дуже високою пропускною здатністю, що робить її ідеальним вибором для середовищ із великим обсягом трафіку, таких як корпоративні мережі або центри обробки даних.

Suricata підтримує глибокий аналіз пакетів (DPI), що дозволяє їй детально вивчати вміст мережевого трафіку, а не тільки його заголовки. Це особливо

важливо для виявлення складних атак, які можуть бути приховані в даних, що передаються в зашифрованому або стиснутому вигляді. Наприклад, Suricata може виконувати аналіз HTTP-трафіку, ідентифікуючи підозрілі запити або відповіді, що містять потенційно шкідливі дані. Аналіз даних на рівні додатків також дозволяє Suricata працювати з протоколами, які використовуються у веб-додатках, таких як HTTPS, FTP, DNS, SMB, TLS/SSL та багатьох інших.

Окремою функцією Suricata є можливість аналізу файлів, які передаються мережею. Платформа здатна розпізнавати тип файлу, екстрагувати його з трафіку та навіть застосовувати правила для визначення потенційно шкідливого ПЗ або інших аномалій. Наприклад, якщо в мережі передається виконуваний файл (.exe), Suricata може аналізувати його метадані, структуру або порівнювати з відомими сигнатурами шкідливого ПЗ. Це розширює можливості платформи за межі стандартного аналізу пакетів, дозволяючи ідентифікувати загрози ще до їх виконання на кінцевих пристроях.

Suricata також інтегрується з популярними системами управління інформацією та подіями безпеки (SIEM), такими як Splunk, Elastic Stack або IBM QRadar. Це забезпечує можливість централізованого управління подіями, аналізу великих обсягів даних та автоматизованого створення звітів про інциденти. Наприклад, Suricata може генерувати журнал подій у форматі JSON, який зручно інтегрується з іншими аналітичними платформами для глибшого аналізу та кореляції подій. Інтеграція з зовнішніми джерелами інформації, такими як IP-репутаційні сервіси або бази даних про вразливості, дозволяє Suricata отримувати актуальні дані про потенційні загрози та адаптувати свої правила виявлення.

Однією з ключових функцій Suricata є підтримка скриптової мови Lua, яка дозволяє користувачам створювати власні правила виявлення та інтегрувати специфічні алгоритми обробки трафіку. Наприклад, за допомогою Lua можна розробити сценарії для аналізу поведінки клієнтів у мережі, моніторингу нестандартних протоколів або виконання специфічних перевірок, які

недоступні у стандартних правилах. Lua-скрипти виконуються безпосередньо на рівні ядра Suricata, що мінімізує затримки та підвищує продуктивність системи.

Suricata також має вбудовану підтримку багаторівневого журналювання, що дозволяє зберігати всі події, пов’язані з мережею, в лог-файли. Це забезпечує детальний аудит та аналіз після інциденту, що є важливим для розслідування атак та покращення заходів безпеки. Наприклад, логи Suricata можна використовувати для виявлення патернів атак або аналізу векторів, які використовували зловмисники.

Suricata надає широкі можливості налаштування для специфічних потреб організацій. Конфігураційний файл suricata.yaml дозволяє детально налаштовувати всі аспекти роботи системи, включаючи джерела мережевого трафіку, формат виводу журналів, порядок обробки правил та інтеграцію з іншими системами. Наприклад, можна налаштувати Suricata для моніторингу окремих VLAN, аналізу зашифрованого трафіку або роботи в режимі "зеркала трафіку".

Таким чином, Suricata є високоефективною, масштабованою та адаптивною системою, яка забезпечує комплексний підхід до захисту мереж. Завдяки поєднанню потужних вбудованих функцій, таких як багатопотокова обробка, підтримка Lua та глибокий аналіз пакетів, вона стає ідеальним вибором для організацій, які шукають надійне рішення для виявлення та запобігання сучасним кіберзагрозам.

Suricata дозволяє використовувати Lua-скрипти через конфігураційний файл suricata.yaml, де задається підтримка мови сценаріїв для користувацьких правил. У процесі роботи система може динамічно викликати Lua-скрипти на основі подій або виявлених аномалій у трафіку. Це дозволяє реалізувати більш гнучкі підходи до аналізу даних, які виходять за межі можливостей стандартних сигнатурних правил. Наприклад, Lua може бути використана для

аналізу контенту HTTP-запитів, виявлення SQL-ін'єкцій чи фільтрації запитів на основі специфічних параметрів.

Однією з основних переваг використання Lua в Suricata є можливість обробки нестандартних або нових протоколів, які не підтримуються "з коробки". Розробник може створити Lua-скрипт, який розпізнає структуру пакетів у протоколі, декодує поля та виконує специфічні перевірки для виявлення потенційних загроз. Наприклад, для роботи з власними протоколами у внутрішній корпоративній мережі Lua дозволяє адаптувати Suricata до унікальних вимог.

Suricata є однією з найбільш ефективних платформ для моніторингу мережевого трафіку та виявлення загроз, яка дозволяє масштабувати захист у великих інфраструктурах, включаючи кілька офісів або дата-центрів. Інтеграція Suricata із системами управління та сенсорами в різних сегментах мережі забезпечує централізоване управління подіями безпеки, аналіз даних і автоматизовану реакцію на інциденти. У цьому контексті використання Lua дозволяє налаштовувати роботу Suricata під специфічні потреби кожного сегмента мережі, створюючи користувацькі правила для аналізу трафіку або взаємодії з іншими компонентами безпеки. На рисунку 2.2 зображено архітектуру розгортання Suricata в масштабованій мережі, яка складається з кількох філій (Branch Office) і центрального офісу або дата-центру (HQ/Data Center) (див. рисунок 2.2). У кожній філії розміщені сенсори Suricata, які моніторять трафік і захищають локальні активи (Protected Assets). Усі сенсори підключені через WAN до центрального офісу, де розташована система управління сенсорами (Sensor Management System). Це забезпечує централізоване управління подіями, що генеруються Suricata, та аналіз виявлених загроз.

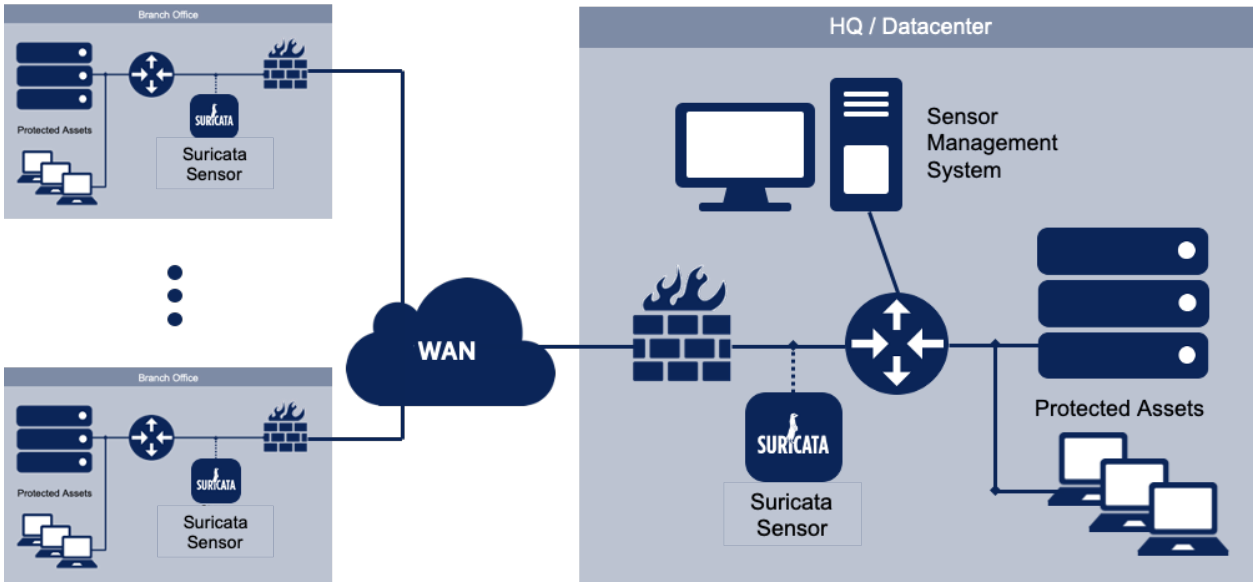

Рисунок 2.2 – Архітектура розгортання Suricata в масштабованій мережі

Використання Lua у такій архітектурі дозволяє створювати скрипти для специфічних задач, які можуть бути різними для кожної філії. Наприклад, у віддаленому офісі Lua-скрипти можуть бути налаштовані для аналізу локальних протоколів, характерних для цього сегмента мережі, або інтеграції з локальними базами даних загроз. У центральному офісі Lua може використовуватися для агрегації даних із сенсорів, створення спеціалізованих логів або інтеграції з SIEM для кореляції подій безпеки.

Lua також може використовуватися для взаємодії з зовнішніми базами даних або API. Наприклад, скрипт може перевіряти IP-адресу джерела запиту у базі репутацій, отримувати інформацію про відомі загрози з публічних API або зберігати дані про загрози у централізованій базі. Це дозволяє інтегрувати Suricata у більш широкі екосистеми кібербезпеки, забезпечуючи її взаємодію з іншими компонентами, такими як SIEM або Threat Intelligence платформи.

Реалізація складних правил із використанням Lua також включає можливість аналізу поведінкових патернів у трафіку. Наприклад, скрипт може визначати відхилення від стандартної поведінки користувачів або клієнтських

пристроїв, виявляти спроби обійти засоби автентифікації або виконувати перевірку на предмет атаки типу "людина посередині".

Однією з переваг Lua є її висока продуктивність. У порівнянні з іншими скриптовими мовами, Lua забезпечує швидке виконання коду, що є важливим для задач реального часу. Це дозволяє використовувати її навіть у високонавантажених середовищах, не створюючи додаткових затримок у роботі системи. Наприклад, обробка HTTP-запитів для ідентифікації шкідливих параметрів може виконуватися з мінімальними затримками, що забезпечує швидке реагування на потенційні загрози.

Крім того, Lua забезпечує можливість створення журналів подій у користувацькому форматі, що дозволяє адаптувати вихідні дані для інтеграції з іншими системами. Наприклад, розробник може створити формат журналів, який відповідає вимогам конкретної SIEM-системи, що значно спрощує обробку даних про загрози.

Окрім згаданих можливостей, варто зазначити, що використання Lua-скриптів у Suricata сприяє створенню цілісної екосистеми захисту, яку можна динамічно розвивати та адаптувати до нових загроз. Lua дозволяє швидко прототипувати нові ідеї і вбудовувати результати аналізу трафіку до більш складних процесів обробки даних. Наприклад, з часом організація може поступово розширювати набір Lua-скриптів, інтегруючи їх з іншими інструментами, такими як системи класифікації та моделювання поведінки користувачів, платформи машинного навчання або спеціалізовані аналітичні сервіси.

Цікавою практикою є застосування Lua для "підшарового" аналізу зашифрованого трафіку. Хоча Suricata не може повністю розшифровувати TLS-потоки без відповідних сертифікатів або зовнішніх механізмів дешифрування, Lua-скрипти можуть аналізувати метадані, такі як послідовність пакетів, часові інтервали між запитами чи розмір і частоту пакетів. Такий підхід допомагає виявляти аномалії у зашифрованому трафіку,

які можуть бути індикаторами обходу стандартних засобів захисту або сигналізувати про командно-контрольні (C2) канали, приховані всередині легітимного HTTPS-потоку.

Ще одна потенційна сфера застосування Lua — інтеграція з інформаційними панелями та платформами візуалізації даних. Розробник може створити Lua-скрипти, які агрегують окремі події Suricata та перетворюють їх у більш «зрозумілий» формат для зовнішніх систем моніторингу. Таким чином, замість масиву сирих подій можна отримати консолідовані індикатори стану мережі, загроз чи ефективності контрзаходів, що значно полегшує роботу команд безпеки.

Для більш наочного розуміння можливостей Lua у Suricata нижче наведено таблицю 2.3, яка ілюструє різні сценарії застосування Lua-скриптів (див. таблицю 2.3):

Таблиця 2.3 – Приклади сценаріїв використання Lua-скриптів у Suricata

Сценарій	Опис	Очікуваний результат
Аналіз нетипових протоколів	Lua-скрипт визначає нестандартну структуру пакетів певного внутрішнього протоколу	Поява можливості виявляти атаки, спрямовані на специфічні, до того невідомі сервіси
Інтеграція з TI-джерелами	Lua звертається до зовнішніх API (Threat Intelligence) для перевірки репутації IP-адрес чи доменів	Оперативне підвищення точності виявлення загроз за рахунок актуальних розвідданих
Поведінковий аналіз аномалій	Lua відстежує часові патерни запитів, глибину взаємодії клієнт-сервер, частоту певних звернень	Виявлення невідомих до цього сценаріїв атак, побудова поведінкових сигнатур
Динамічна адаптація правил	Lua-скрипт вносить зміни до наявних правил залежно від поточної ситуації (сплеску атак або нових IOC)	Зменшення часу реагування на нові загрози, підтримка актуальності політик безпеки
Агрегація та консолідація даних	Lua форматує журнали Suricata для зручного імпорту в системи візуалізації (Grafana, Kibana)	Спрощений аналіз великих обсягів даних, швидша ідентифікація тенденцій та інцидентів

Окремо слід відзначити, що впровадження Lua-скриптів може стати основою для створення "бібліотек" користувацьких рішень, якими можуть

ділитися фахівці з кібербезпеки у межах однієї організації чи навіть спільноти. Це може спростити передачу знань, забезпечити повторне використання вже створених інструментів та пришвидшити реакцію на нові виклики.

Перспективним напрямком розвитку є застосування Lua у поєднанні з методами машинного навчання та інтелектуального аналізу даних. Наприклад, Lua-скрипт може виконувати роль "моста" між Suricata та зовнішньою ML-платформою, передаючи туди агреговані метрики трафіку. У відповідь він може отримувати оцінки ризику або рекомендації щодо фільтрації. Такий підхід дозволяє поєднати переваги ручної логіки (Lua) з адаптивністю машинного навчання, що особливо актуально у динамічних умовах сучасних кіберзагроз.

Таким чином, використання Lua-скриптів для розширення можливостей Suricata не обмежується простим додаванням користувацьких правил. Це цілий комплекс практик, які дозволяють гнучко адаптувати систему до специфічних умов, цілей та загроз, інтегрувати її у складні екосистеми безпеки, підвищувати ефективність аналізу трафіку, а також створювати фундамент для подальшої інноваційної еволюції захисних механізмів.

2.3 Методологічні основи інтеграції Lua з іншими компонентами кібербезпеки (Threat Intelligence, SIEM та SOAR)

Методологічні основи інтеграції Lua з іншими компонентами кібербезпеки полягають у виробленні єдиного концептуального підходу до розробки, впровадження та експлуатації рішень, які розширюють можливості систем виявлення та запобігання вторгнень (IDS/IPS). Використання Lua як інтеграційного інструмента дозволяє динамічно адаптувати функціонал IDS/IPS відповідно до нових загроз, залучати зовнішні джерела інформації, розширювати контекст аналізу подій безпеки та автоматизувати процеси реагування.

Методологія інтеграції Lua з платформами Threat Intelligence (TI) спрямована на оперативне отримання актуальних даних про потенційні загрози. Наприклад, IDS/IPS можуть використовувати Lua-скрипти для звернення до зовнішніх баз даних загроз, отримуючи інформацію про шкідливі IP-адреси, домени чи сигнатури атак. Застосування цієї інформації у реальному часі дає змогу не лише виявляти відомі типи загроз, а й швидко реагувати на нові, що тільки з'явилися. Таким чином, IDS/IPS переходять від статичного підходу до динамічного, враховуючи постійно оновлювану інформацію та підвищуючи точність і релевантність прийнятих рішень [16].

Інтеграція з системами управління інформацією та подіями безпеки (SIEM) надає можливість Lua-скриптам формувати та передавати події в узгоджених форматах, таких як JSON чи CEF. Завдяки цьому SIEM отримує деталізовані, збагачені контекстом події, що надходять від IDS/IPS. Якщо IDS/IPS виявляє підозрілу активність, Lua може додатково верифікувати її, звертаючись до TI-джерел або локальних баз даних, і вже потім надсилати подію до SIEM. Такий підхід підвищує якість даних, що надходять у SIEM, знижуючи кількість хибних спрацьовувань і дозволяючи аналітикам безпеки зосередитися на найбільш важливих інцидентах. Крім того, можливість динамічного керування форматом та повнотою подій дає змогу оперативно адаптувати IDS/IPS до змін у вимогах SIEM чи особливостях конкретної організації.

Інтеграція з платформами оркестрації, автоматизації та реагування на інциденти (SOAR) розширює можливості Lua ще далі. SOAR відповідає за виконання заздалегідь визначених сценаріїв реагування (плейбуків), які можуть включати автоматичне блокування підозрілого трафіку, ізоляцію ураженої системи або надсилання сповіщень команді безпеки. Lua-скрипти в IDS/IPS можуть виступати ініціаторами запуску таких плейбуків, передаючи в SOAR контекст інциденту (наприклад, IP-адресу джерела атаки, сигнатуру загрози чи тип підозрілої діяльності). Таким чином, система отримує

двосторонній зв’язок: IDS/IPS, підкріплені Lua-скриптами, не тільки генерують сигнали для SOAR, а й можуть отримувати зворотні інструкції про зміну правил виявлення чи пріоритетів, залежно від результатів автоматизованого аналізу. Це забезпечує гнучку й адаптивну модель реагування на інциденти, де дії налаштовуються з урахуванням загального контексту безпеки.

Методологічні основи інтеграції Lua з іншими компонентами кібербезпеки також передбачають дотримання належних практик розробки та впровадження. Використання стандартизованих форматів даних (JSON, YAML, STIX/TAXII) і загальноприйнятих протоколів (HTTPS, REST API, Syslog) сприяє інтероперабельності з різними платформами та вендорами. Необхідно враховувати надійність: Lua-скрипти мають адекватно обробляти ситуації, коли зовнішні джерела недоступні, а SIEM чи SOAR не реагують. Оптимізація продуктивності, кешування запитів, можливість роботи в умовах деградації — усе це забезпечує стійкість та масштабованість інтегрованої системи безпеки.

Важливими складовими методології є версіонування коду та документація. Lua-скрипти, які реалізують інтеграцію, слід зберігати у репозиторіях контролю версій, описувати їхню функціональність, формати вхідних та вихідних даних, а також процедури оновлення та тестування. Це дозволить швидко адаптуватися до змін інфраструктури або вимог безпеки, забезпечуючи чіткість та прозорість процесів.

Зрештою, інтеграція Lua з TI, SIEM та SOAR формує цілісний механізм, у якому IDS/IPS перестають бути ізольованими елементами. Вони перетворюються на розумних агентів у розгалуженій екосистемі кібербезпеки, здатних не лише виявляти загрози, а й узгоджувати свої дії з урахуванням зовнішніх даних, контексту подій і рекомендацій автоматизованих систем реагування. Це суттєво підвищує загальний рівень захищеності організації,

дозволяючи ефективно використовувати наявні ресурси та оперативно реагувати на нові виклики.

Одним із центральних елементів розробки методології інтеграції Lua з TI, SIEM та SOAR є створення чіткої архітектурної моделі, яка формалізує точки взаємодії між IDS/IPS та іншими компонентами. Така модель повинна відображати логічні етапи обробки даних, можливі сценарії обміну інформацією, а також умови переходу від одного етапу до іншого. Формальний опис цих процесів дозволить команді кібербезпеки чітко зрозуміти, які саме функції виконують Lua-скрипти, коли вони запускаються, які дані передаються, в якому форматі і яким чином перевіряється їх достовірність [17].

Важливим аспектом є впровадження "прослуховувальників" подій (event listeners) у IDS/IPS. Ці механізми дозволяють Lua-скриптам реагувати на широкий спектр ситуацій:

–коли IDS/IPS виявляє новий тип аномального трафіку;

–коли SIEM надсилає зворотний сигнал про кореляцію кількох інцидентів в одну складну атаку;

–коли TI-джерело оновлює репутаційні дані і необхідно терміново змінити локальні політики безпеки;

–коли SOAR ініціює масштабну зміну конфігурації у відповідь на нову глобальну загрозу.

У таких випадках Lua-скрипти можуть динамічно регулювати поведінку IDS/IPS:

–оновлювати набір правил виявлення;

–змінювати рівні чутливості для певних сигнатур;

–інтегрувати отримані TI-дані безпосередньо в логіку виявлення;

–передавати у SIEM лише події, що мають високий пріоритет;

–запускати в SOAR ланцюжки реакцій у міру появи нових контекстуальних даних.

Для кращого розуміння можливих сценаріїв застосування Lua у межах інтегрованої екосистеми безпеки доцільно розглянути узагальнений приклад використання скриптів у взаємодії з TI, SIEM та SOAR. Нижче наведена таблиця 2.4, яка ілюструє типові ситуації, в яких Lua виступає сполучною ланкою між IDS/IPS та зовнішніми інструментами. Вона показує, як гнучке налаштування логіки Lua дозволяє швидко реагувати на зміни в кіберсередовищі, оптимізувати робоче навантаження на аналітичні платформи, а також забезпечувати безперервну адаптацію політик безпеки (див. таблицю 2.4).

Таблиця 2.4 – Приклади сценаріїв взаємодії Lua з компонентами кібербезпеки

Компонент	Ситуація	Дія Lua	Результат
TI	TI недоступний	Використати кешовані дані	IDS/IPS не втрачає ефективність
TI	Новий список шкідливих доменів	Оновити локальні сигнатури	Швидка реакція на нові загрози
SIEM	Надмірна кількість подій	Фільтрувати малозначимі події	Зниження навантаження на SIEM
SIEM	Отримано сигнал про складну кореляцію	Додати контекст у наступних подіях	Краща інформованість аналітиків
SOAR	Виявлена критична загроза	Ініціювати блокування трафіку	Автоматизоване миттєве реагування
SOAR	Запит на зміну політик від SOAR	Налаштувати нові правила і рівні чутливості	IDS/IPS гнучко підлаштовується під контекст

Розглянемо фрагмент коду на Lua, що демонструє практичний підхід до інтеграції з TI та SIEM. Цей приклад підкреслює роль Lua як інструмента, що не лише проводить первинний аналіз трафіку та оновлює локальні правила, але й здатен активно залучати зовнішні джерела даних, а також ініціювати дії у відповідь через SOAR. Застосування скриптів дозволяє підвищити точність виявлення загроз, покращити якість даних, що надходять до SIEM, та прискорити реагування на інциденти. Такий програмний підхід має бути

інтегрований у цілісну методологію, де процеси розробки, тестування та підтримки коду Lua чітко регламентовані, а обмін інформацією з іншими компонентами кібербезпеки відбувається за узгодженими правилами та форматами.

Лістинг 2.1 – Приклад коду Lua, що ілюструє взаємодію з SIEM та TI

```
-- Припустимо, ми маємо функції для роботи з HTTP та JSON
local http = require("http")

function update_rules_from_ti(ti_url, token)
local resp = http.get(ti_url, { headers = { ["Authorization"] = "Bearer " .. token } })
if resp.status == 200 then
local data = parse_json(resp.body)
for _, entry in ipairs(data.bad_ips) do
add_to_blacklist(entry.ip)
end
else
-- Якщо TI недоступний – працюємо з кешем
local cached_data = load_cached_ti_data()
for _, cached_ip in ipairs(cached_data.bad_ips or {}) do
add_to_blacklist(cached_ip)
end
end
end

function send_event_to_siem(event_url, event_data)
local headers = { ["Content-Type"] = "application/json" }
local body = to_json(event_data)
local resp = http.post(event_url, headers, body)
return resp.status == 200
end

function on_suspicious_activity(ip, details)
```

Продовження лістингу 2.1

```
-- Запит репутації
local score, level = check_ip_reputation(ip)
if score > 80 then
local event = {
source_ip = ip,
severity = "critical",
message = "High risk IP detected with level: " .. level,
details = details
}
send_event_to_siem("https://siem.example.com/input", event)
run_soar_playbook("block_ip", { ip = ip })
end
end
```

У цьому прикладі Lua-скрипт:

– звертається до TI для оновлення правил, використовуючи кеш у разі недоступності сервісу;

– надсилає події у SIEM з обраним форматом;

– ініціює запуск плейбуку в SOAR при виявленні критичної загрози.

Важливо зазначити, що методологія інтеграції Lua з TI, SIEM та SOAR повинна включати питання безпеки самих скриптів. Наприклад, слід передбачити:

–механізми ізоляції (sandboxing) Lua-коду, щоб помилки або шкідливі сценарії не могли компрометувати IDS/IPS;

–перевірку цілісності скриптів та контроль доступу до редагування коду;

–регламент тестування змін у тестовому середовищі перед релізом у бойове оточення.

Крім того, методологія має описувати процес ескалації інцидентів. Якщо Lua-скрипт виявив загрозу та надіслав подію у SIEM чи запустив плейбук у SOAR, важливо мати чіткі вказівки щодо подальших дій персоналу,

моніторингу результатів автоматизованого реагування та коригування політик безпеки на основі зворотного зв’язку.

Зрештою, інтеграція Lua з компонентами кібербезпеки створює гнучке, адаптивне середовище, де IDS/IPS перетворюється на динамічний вузол у розгалуженій екосистемі захисту. Правильно розроблена методологія дозволяє використати потенціал Lua для підвищення ефективності, масштабованості та релевантності заходів безпеки, водночас мінімізуючи ризики, пов’язані з помилками у сценаріях, перевантаженням сервісів чи несвоєчасною реакцією на нові загрози.

РОЗДІЛ 3. РОЗРОБКА ПРАКТИЧНОГО РІШЕННЯ РОЗШИРЕННЯ IDS/IPS ЗА ДОПОМОГОЮ LUA

3.1 Встановлення та конфігурація Suricata.

Розгортання віртуального середовища є важливим етапом у підготовці платформи для розробки та тестування рішень, пов'язаних з використанням мови Lua для розширення функціональних можливостей IDS/IPS. Використання віртуальних машин забезпечує гнучкість, ізольованість та безпеку експериментів, дозволяючи уникнути негативного впливу на основну операційну систему або інші критичні сервіси.

Для створення робочого середовища використання VMware Workstation Player є одним із найзручніших варіантів (див. рисунок 3.1). Цей програмний продукт забезпечує простий і інтуїтивний інтерфейс управління віртуальними машинами, підтримує широкий спектр операційних систем, а також надає можливість детально налаштовувати апаратні ресурси та мережеву конфігурацію. Обрання саме VMware зумовлене його стабільністю, поширеністю та доброю сумісністю з різними операційними системами [18].

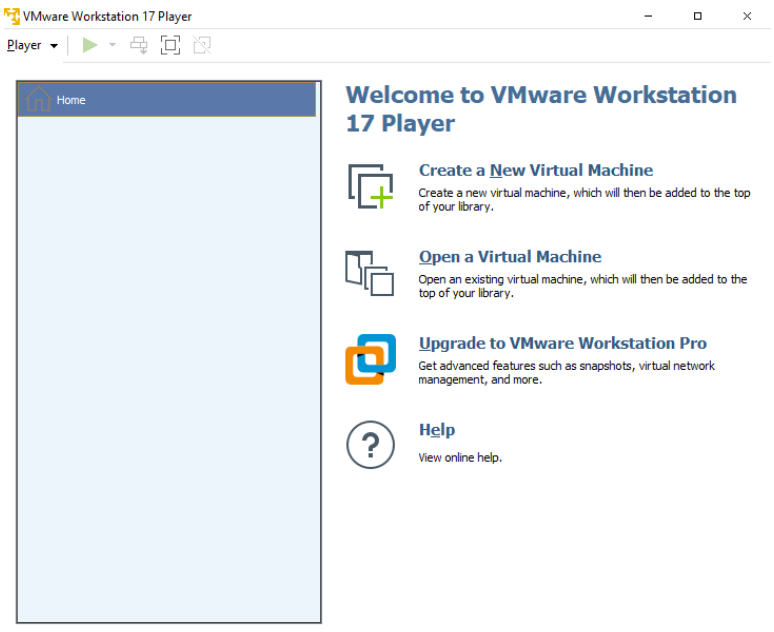

Рисунок 3.1 – Інтерфейс VMware Workstation Player 17

Для реалізації поставлених завдань доцільно використати операційну систему Ubuntu LTS. Цей дистрибутив Linux має такі переваги, як стабільність, регулярні оновлення безпеки, широку базу пакетів для мережевого аналізу, зокрема для встановлення Suricata та Lua, а також велику спільноту користувачів і розробників, здатну надати підтримку в разі виникнення проблем. Довгострокова підтримка (LTS) забезпечує отримання оновлень безпеки протягом тривалого часу, що важливо для захисту від потенційних вразливостей під час експериментів.

Спершу необхідно мати два ключових компоненти: встановлений VMware Workstation Player та ISO-образ дистрибутиву Ubuntu. Якщо VMware Workstation Player ще не встановлено, слід завантажити останню версію з офіційного сайту виробника та виконати інсталяцію відповідно до інструкцій. Після цього потрібно завантажити ISO-образ Ubuntu LTS, перейшовши на офіційний сайт ubuntu.com і обравши відповідний розділ завантажень (див. рисунок 3.2). На момент виконання цієї роботи актуальною LTS-версією є Ubuntu 24.04 [19].

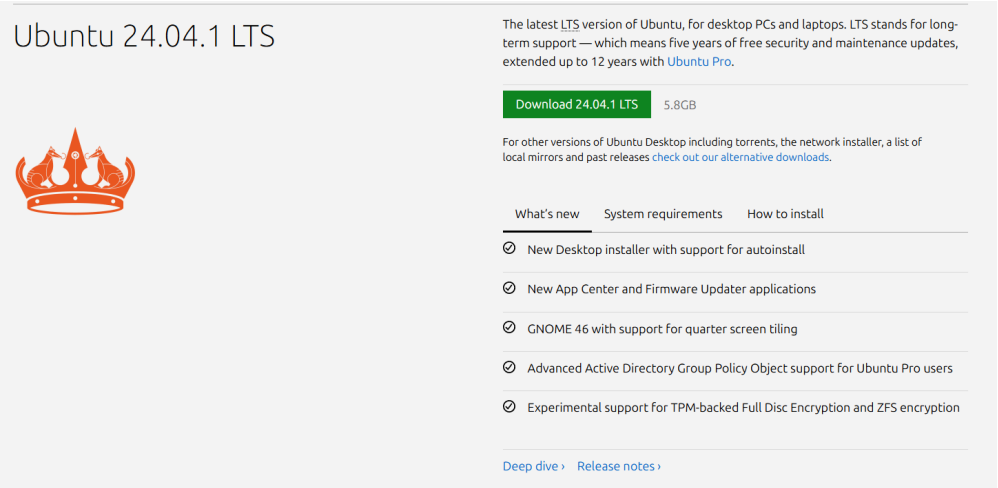

Рисунок 3.2 – Сторінка завантаження ISO-образу Ubuntu 24.04

Після завантаження ISO-образу Ubuntu переходимо до створення нової віртуальної машини у VMware Workstation Player. Запускаємо програму та обираємо опцію створення нової віртуальної машини. Далі:

–вибираємо режим, при якому інсталяція ОС буде виконана з ISO-образу, але спершу обираємо опцію самостійного вказання образу пізніше (I will install the operating system later).

–у списку операційних систем обираємо Linux, а у версіях – Ubuntu 64-bit

–задаємо ім’я віртуальної машини, наприклад "Ubuntu LTS IDS/IPS", та вказуємо каталог, де будуть зберігатися файли віртуальної машини.

–налаштовуємо розмір віртуального диска. Для експериментів з Suricata та Lua доцільно виділити принаймні 20-30 ГБ, залежно від наявних ресурсів. Варто обрати динамічне виділення простору, що дозволить не займати одразу весь обсяг диска

–після створення віртуальної машини відкриваємо її налаштування та в розділі CD/DVD Drive вказуємо шлях до завантаженого ISO-образу Ubuntu

Закінчивши ці кроки, запускаємо створену віртуальну машину. Під час запуску з’явиться меню завантаження, де можна обрати опцію "Try or Install Ubuntu" або одразу перейти до інсталяції.

Після завантаження інсталятора обираємо бажану мову інтерфейсу, розкладку клавіатури, конфігурацію мережі. При встановленні Ubuntu можна залишити більшість параметрів за замовчуванням, якщо немає специфічних вимог.

Далі буде запропоновано виконати "Erase disk and install Ubuntu" – ця дія стосується лише віртуального диска віртуальної машини і ніяк не вплине на реальну систему хоста. Підтверджуємо встановлення, створюємо ім’я користувача і пароль (див. рисунок 3.3), обираємо часовий пояс. Потім чекаємо завершення встановлення, що зазвичай займає декілька хвилин.

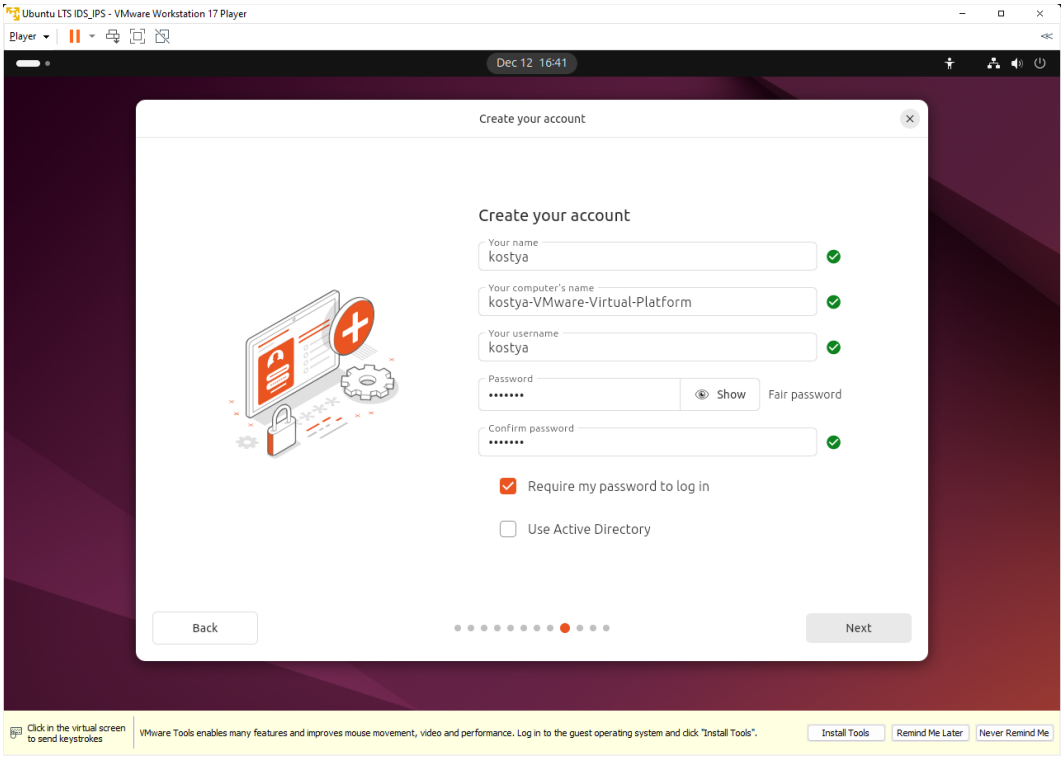

Рисунок 3.3 – Створюємо користувача і встановлюємо пароль

Після завершення інсталяції необхідно перезавантажити віртуальну машину, вийнявши встановлювальний носій (ISO-образ) зі списку пристроїв або просто натиснувши Enter, як буде запропоновано. Після перезавантаження користувач потрапляє до робочого столу Ubuntu, де можна виконувати всі необхідні дії.

Таким чином, у результаті виконання описаних кроків ми матимемо готову віртуальну машину з операційною системою Ubuntu LTS у середовищі VMware Workstation Player. Це середовище стане основою для подальшого встановлення Suricata, розробки та впровадження Lua-скриптів, а також проведення експериментів з розширення функціональних можливостей IDS/IPS без ризику пошкодження основної системи.

Після успішного встановлення та налаштування віртуальної машини з Ubuntu ми можемо перейти до встановлення Suricata – системи виявлення та запобігання вторгнень, яка буде основою для інтеграції Lua-скриптів. Suricata підтримує офіційний PPA-репозиторій для Ubuntu, що спростить процес

інсталяції актуальної стабільної версії. Додатково ми встановимо інструмент jq, який стане в пригоді для зручного перегляду та аналізу JSON-виводу подій, що генерує Suricata.

Спершу виконаємо встановлення необхідних пакетів (див. рисунок 3.4):

sudo apt-get install software-properties-common

sudo add-apt-repository ppa:oisf/suricata-stable

sudo apt update

sudo apt install suricata jq

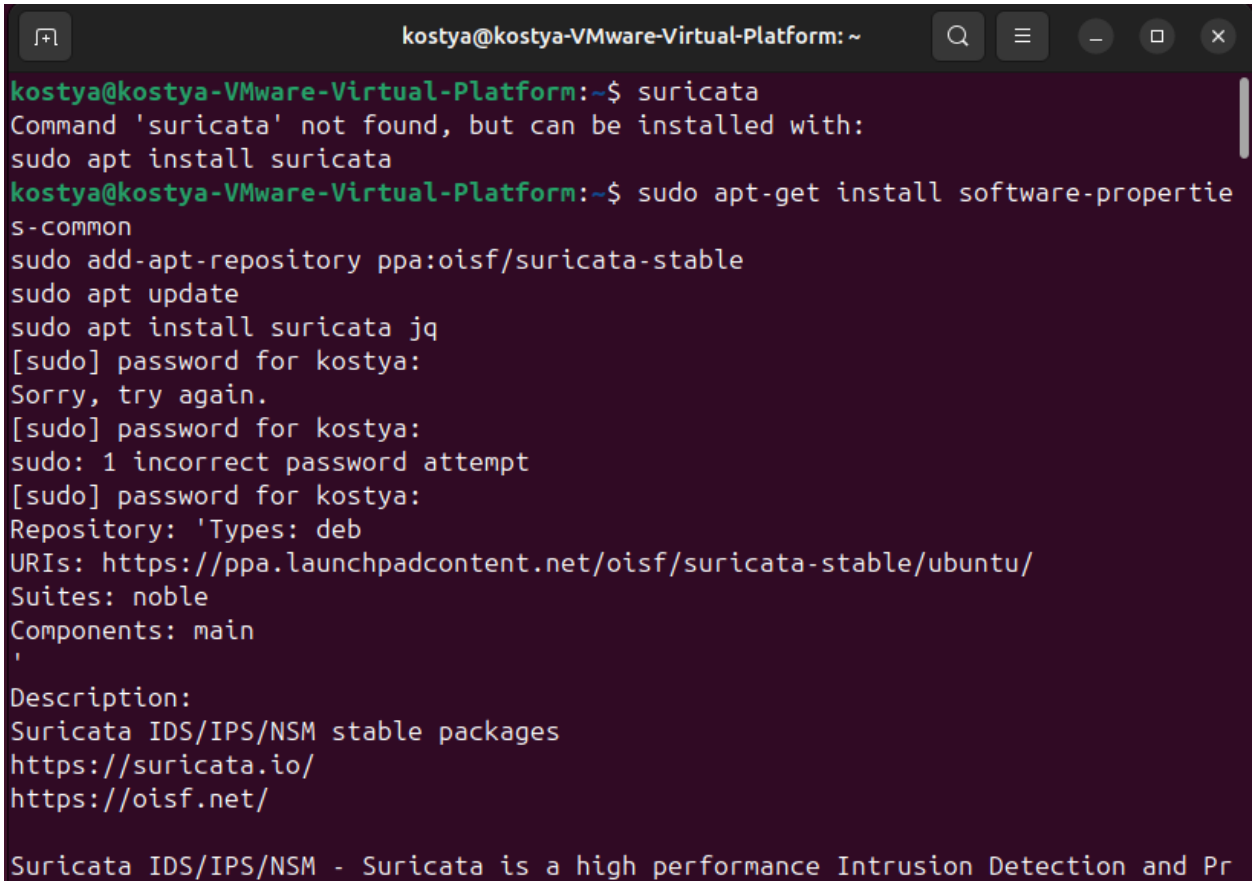

Рисунок 3.4 – Встановлення необхідних пакетів Suricata

Після цього Suricata буде встановлена разом із jq. Для перевірки версії та параметрів збірки Suricata можна скористатись наступною командою (див. рисунок 3.5): sudo suricata --build-info

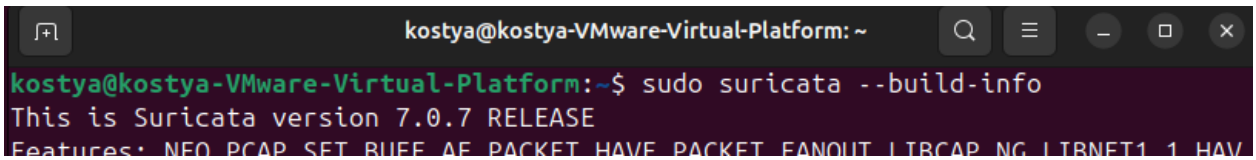

Рисунок 3.5 – Перевірка версії збірки Suricata

Це допоможе переконатися, що Suricata встановлена коректно. Також можна перевірити стан служби (див. рисунок 3.6): sudo systemctl status suricata

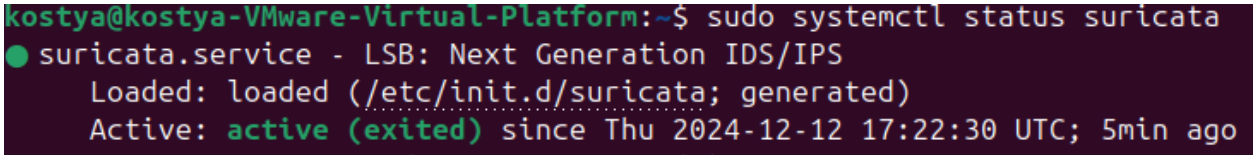

Рисунок 3.6 – Перевірка стану служби Suricata

На цьому етапі варто визначити мережевий інтерфейс та IP-адресу, з якими працюватиме Suricata. Для перегляду доступних інтерфейсів у системі використовується команда: ip addr. Вивід команди покаже список інтерфейсів та їх IP-адреси (див. рисунок 3.7). Наприклад, може бути виявлено інтерфейс:

2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 ...
link/ether 00:0c:29:5d:a8:b4 ...
inet 192.168.175.128/24 brd 192.168.175.255 scope global dynamic ...

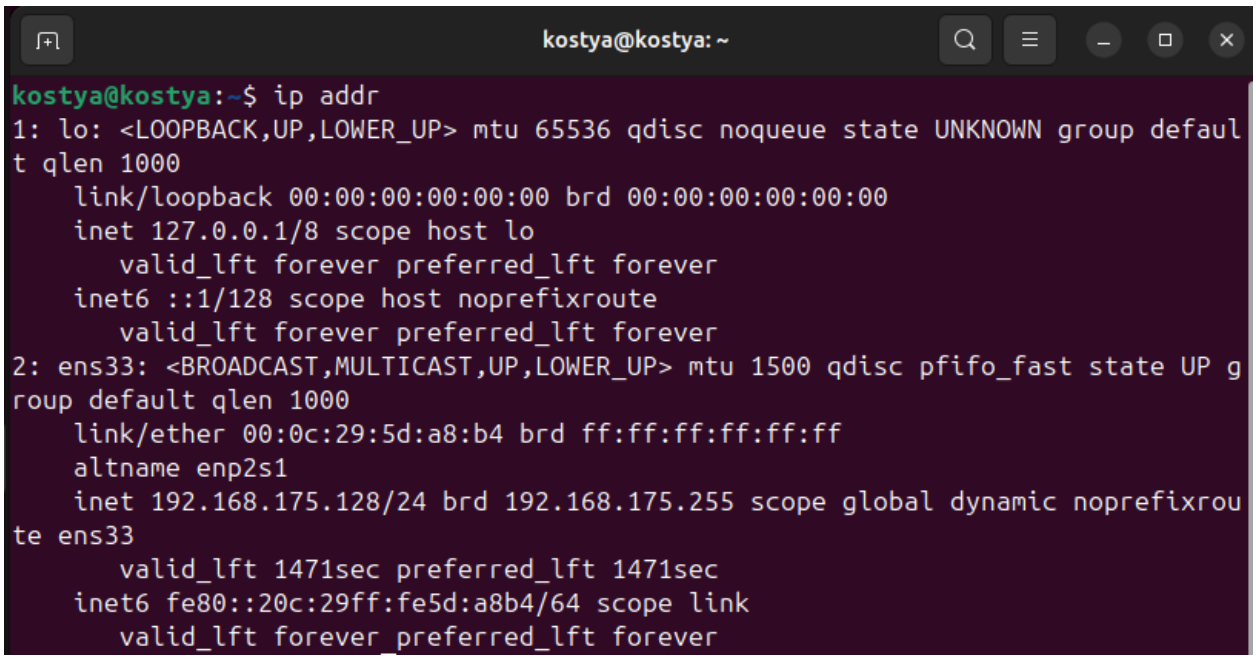

Рисунок 3.7 – Список інтерфейсів та IP-адрес

Оскільки реальний інтерфейс має назву ens33, а підмережа – 192.168.175.128/24, саме ці дані і слід використовувати у конфігурації, відкриваємо конфігураційний файл Suricata для налаштування параметрів, включаючи змінну HOME_NET та інтерфейси: sudo vim /etc/suricata/suricata.yaml

Змінна HOME_NET за замовчуванням включає адреси RFC 1918 (приватні мережі), але її слід уточнити під власне середовище. В нашому випадку використовується підмережа 192.168.175.128/24, можна зазначити: HOME_NET: "[192.168.175.128/24]" (див. рисунок 3.8).

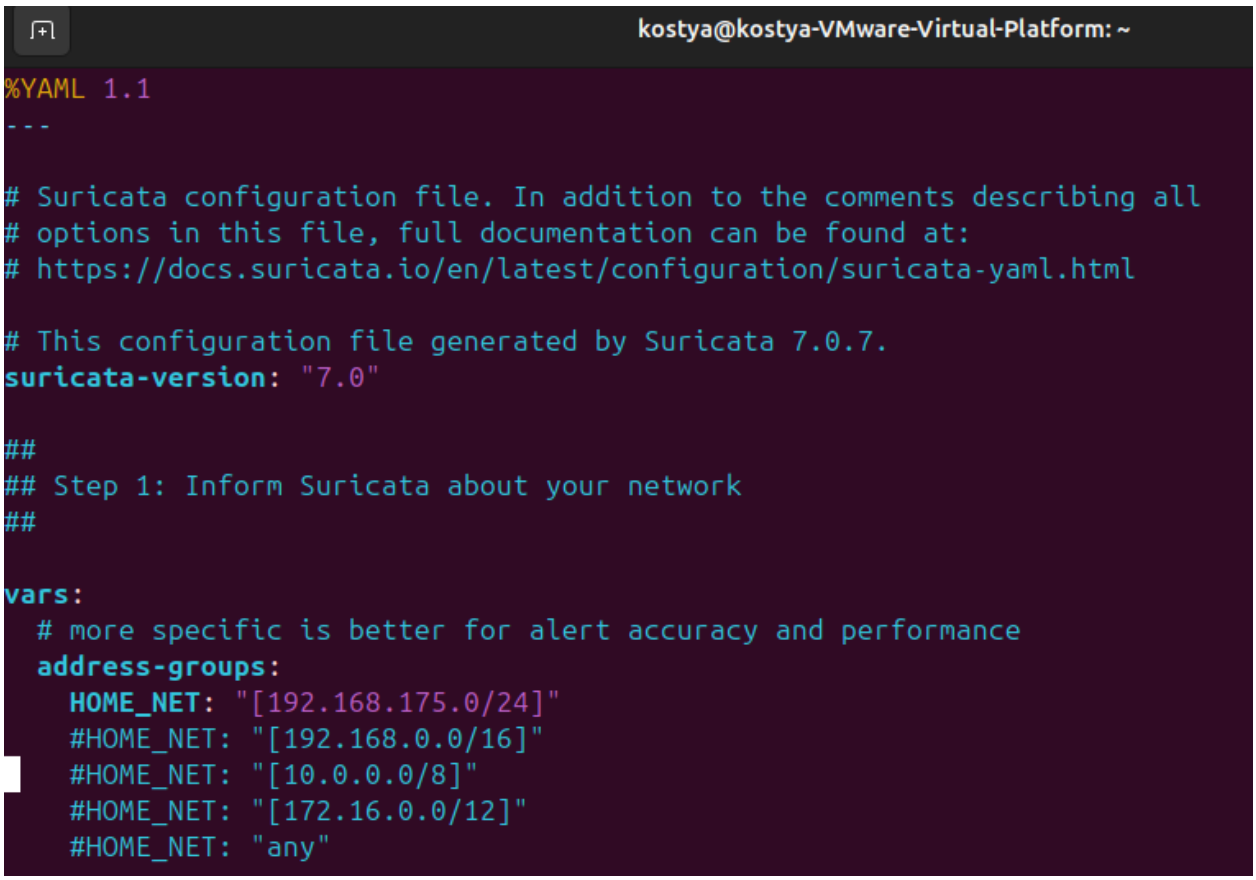

Рисунок 3.8 – Налаштування HOME_NET

У розділі налаштувань af-packet замінюємо прикладове ім’я інтерфейсу на ens33 (див. рисунок 3.9): af-packet:

- interface: ens33

  cluster-id: 99

  cluster-type: cluster_flow

  defrag: yes

  use-mmap: yes

  tpacket-v3: yes

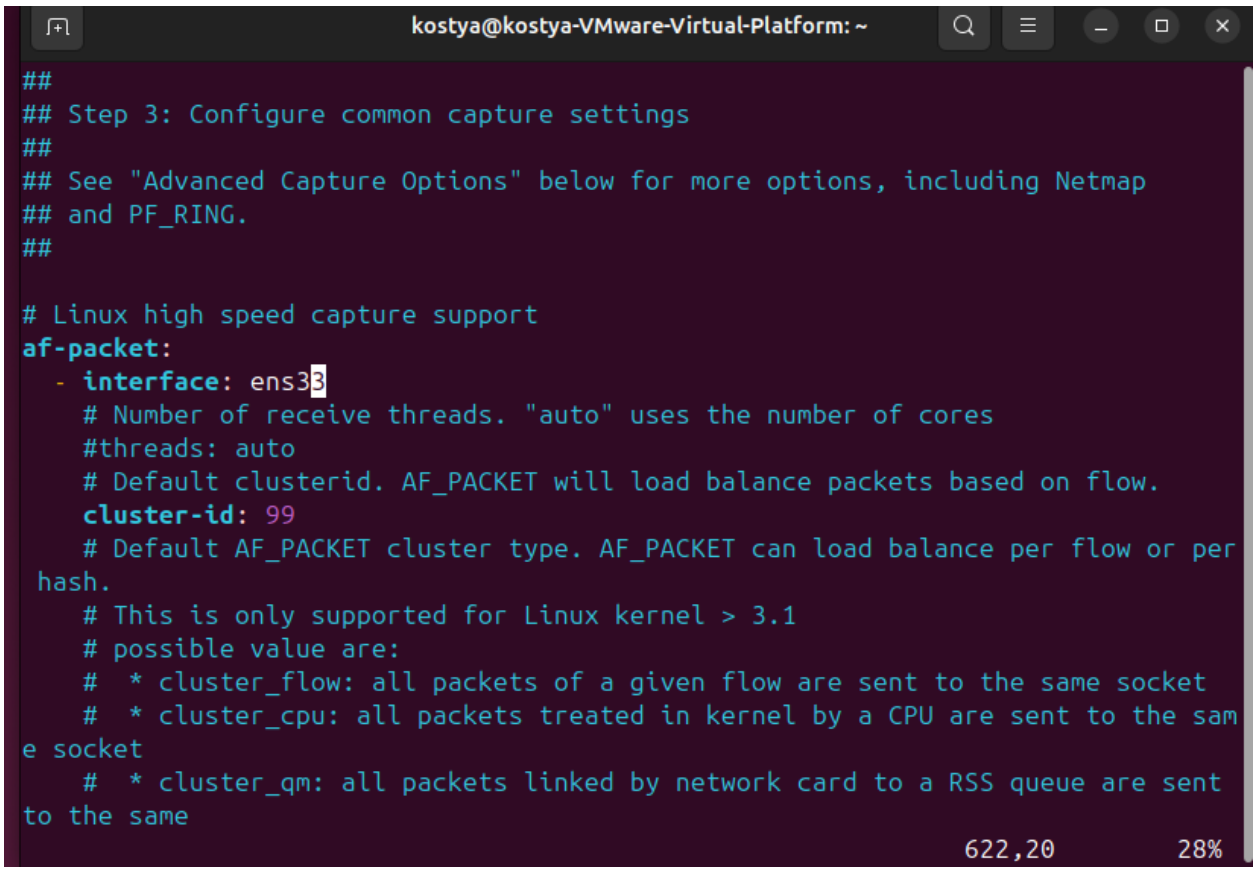

Рисунок 3.9 – Зміна ім’я інтерфейсу

Також у файлі suricata.yaml необхідно знайти розділ security та переконатися, що в ньому активована можливість використання Lua-скриптів у правилах. Це можна зробити, встановивши параметр allow-rules у значення true. Конфігурація цього блоку має виглядати наступним чином (див. рисунок 3.10):

```
lua:
  # Allow Lua rules. Disabled by default.
  allow-rules: true
```

Рисунок 3.10 – Активування можливості використання Lua-скриптів у правилах

У файлі suricata.yaml також необхідно налаштувати секцію, яка відповідає за підтримку Lua-скриптів для генерації алертів і подій. Це робиться у розділі

outputs, де потрібно активувати Lua Output Support, встановивши параметр enabled у значення yes. Крім того, у цій секції слід вказати директорію, в якій зберігаються скрипти Lua (scripts-dir), а також назви скриптів, які будуть використовуватися (див. рисунок 3.11).

```
# Lua Output Support - execute lua script to generate alert and event
# output.
# Documented at:
# https://docs.suricata.io/en/latest/output/lua-output.html
- lua:
    enabled: yes
    scripts-dir: /etc/suricata/lua-output/
    scripts:
        - portscanDetection.lua
       #- check_request.lua
       #- test_http.lua
       # - http_log.lua
```

Рисунок 3.11 – Активування підтримки Lua скриптів

Це налаштування дозволяє Suricata виконувати Lua-скрипти з зазначеної директорії /etc/suricata/lua-output/, забезпечуючи гнучкість у визначенні алертів та подій. У даному прикладі активовано скрипт portscanDetection.lua, який відповідає за виявлення порт-сканування. Інші скрипти (check_request.lua, test_http.lua, http_log.lua) вказані для прикладу і можуть бути використані при необхідності, якщо їх закоментовані рядки будуть розкоментовані.

Таким чином ми адаптували конфігурацію під реальну мережеву інфраструктуру користувача.

Після внесення змін зберігаємо файл та встановлюємо правила Suricata за допомогою: sudo suricata-update

За замовчуванням буде завантажено набір ET Open rules, збережений у /var/lib/suricata/rules. Потім перезапускаємо Suricata, щоб застосувати конфігурацію та правила: sudo systemctl restart suricata. Для перевірки коректності роботи переглядаємо лог: sudo tail /var/log/suricata/suricata.log.

Якщо все налаштовано правильно, у логу будуть повідомлення про успішний запуск потоку обробки пакетів (див. рисунок 3.12).

```
kostya@kostya-VMware-Virtual-Platform:~$ sudo systemctl restart suricata
kostya@kostya-VMware-Virtual-Platform:~$ sudo tail /var/log/suricata/suricata.log
[5337 - Suricata-Main] 2024-12-12 21:44:53 Perf: detect: AppLayer MPM "toserver file_data (ftp)": 17
[5337 - Suricata-Main] 2024-12-12 21:44:53 Perf: detect: AppLayer MPM "toclient file_data (ftp-data)": 17
[5337 - Suricata-Main] 2024-12-12 21:44:53 Perf: detect: AppLayer MPM "toserver file_data (ftp-data)": 17
[5337 - Suricata-Main] 2024-12-12 21:44:53 Perf: detect: AppLayer MPM "toclient file_data (http)": 17
[5337 - Suricata-Main] 2024-12-12 21:44:53 Perf: detect: AppLayer MPM "toserver file_data (http)": 17
[5337 - Suricata-Main] 2024-12-12 21:44:53 Perf: detect: AppLayer MPM "toclient file_data (http2)": 17
[5337 - Suricata-Main] 2024-12-12 21:44:53 Perf: detect: AppLayer MPM "toserver file_data (http2)": 17
[5337 - Suricata-Main] 2024-12-12 21:44:53 Perf: detect: AppLayer MPM "toserver file_data (smtp)": 17
[5337 - Suricata-Main] 2024-12-12 21:44:53 Perf: detect: Pkt MPM "icmpv6.hdr": 1
[5337 - Suricata-Main] 2024-12-12 21:44:53 Perf: detect: Pkt MPM "ipv6.hdr": 1
```

Рисунок 3.12 – Успішний запуск потоку обробки пакетів

Для тестування можна скористатись вже наявною тестовою сигнатурою. Перед виконанням тесту слідкуємо за файлами журналів оповіщень:

sudo tail -f /var/log/suricata/fast.log

curl http://testmynids.org/uid/index.html

Після виконання запиту у файлі fast.log з’явиться сповіщення про спрацювання правила. Для більш детального аналізу використовується формат EVE JSON та інструмент jq, наприклад (див. рисунок 3.13): sudo tail -f /var/log/suricata/eve.json | jq 'select(.event_type=="alert")'

```
kostya@kostya-VMware-Virtual-Platform:~$ sudo tail -f /var/log/suricata/eve.json | jq 'select(.event_type=
="alert")'
{
  "timestamp": "2024-12-12T21:48:25.677455+0000",
  "flow_id": 539916065639231,
  "in_iface": "ens33",
  "event_type": "alert",
  "src_ip": "18.244.146.117",
  "src_port": 80,
  "dest_ip": "192.168.175.128",
  "dest_port": 56172,
  "proto": "TCP",
  "pkt_src": "wire/pcap",
  "tx_id": 0,
  "alert": {
    "action": "allowed",
```

Рисунок 3.13 – Перегляд логів з використанням JSON формату

Таким чином, базове налаштування Suricata завершено з урахуванням реальних даних: інтерфейс ens33 та підмережа 192.168.175.128/24. Тепер можна переходити до інтеграції Lua-скриптів, створення користувацьких правил та детальнішого аналізу трафіку.

3.2 Розробка та тестування Lua-скриптів для виявлення загроз.

Для початку розглянемо, як працюють правила. Правила в Suricata є основою для аналізу мережевого трафіку та виявлення потенційних загроз. Вони визначають, яку поведінку в мережі необхідно відстежувати, а також які дії виконувати у відповідь на цю поведінку. Кожне правило містить кілька обов'язкових елементів, які разом визначають його функціональність. На рисунку 3.14 представлено приклад правила з поясненням кожного його елемента (див. рисунок 3.14).

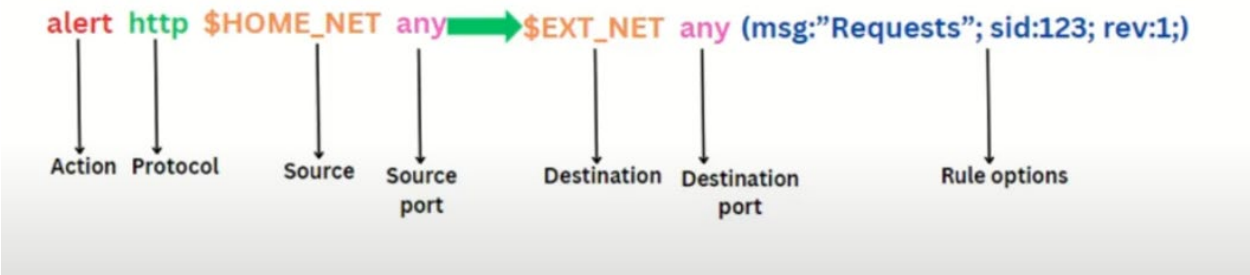

Рисунок 3.14 – Структура правила в Suricata

У цьому прикладі показано правило: alert http $HOME_NET any -> $EXT_NET any (msg:"Requests"; sid:123; rev:1;)

Розбір структури правила:

–action (alert) – Дія, яку Suricata виконує при виконанні умови правила. У цьому випадку alert означає створення сповіщення.

–protocol (http) – Протокол, який аналізується (наприклад, HTTP, TCP, UDP або ICMP).

–source ($HOME_NET) – Джерело трафіку. Зазвичай визначається за допомогою змінних, таких як $HOME_NET (внутрішня мережа).

–source Port (any) – Порт джерела. У цьому випадку any означає, що порт може бути будь-яким.

–destination ($EXT_NET) – Цільовий вузол або мережа. $EXT_NET зазвичай представляє зовнішню мережу.

–destination Port (any) – Порт призначення. Як і у випадку з джерелом, any означає будь-який порт.

–rule Options (msg, sid, rev) – Опції правила, які описують його функціональність:

–msg – Текст повідомлення, що буде відображатися у журналі.

–sid – Унікальний ідентифікатор правила.

–rev – Ревізія правила, що використовується для відстеження його змін.

Ключовим аспектом правил у Suricata є використання змінних, таких як $HOME_NET та $EXT_NET, що дозволяє легко адаптувати їх до різних середовищ. У нашому випадку змінна $HOME_NET визначена як 192.168.175.128/24.

Розроблений алерт має наступну структуру (див. лістинг 3.1):

Лістинг 3.1 – Структура алерта

```
alert tcp any any -> any any (msg:"Port Scan Detected (portscanDetection.lua) : Over 9 different ports in 120 sec."; priority:2; flow:to_server; threshold: type limit, track by_src, seconds 1200, count 1; lua:portscanDetection.lua; sid:1000001; rev:1;)
```

Цей алерт є прикладом правила, яке використовує Lua-скрипт для виявлення спроб сканування портів у мережевому трафіку.

Почнемо з основних елементів правила. Ключове слово alert визначає тип дії, яку Suricata має виконати, коли умови правила будуть задоволені. У цьому випадку, alert вказує на створення сповіщення про виявлену загрозу.

Протокол, який аналізується, зазначено як tcp. Це означає, що правило буде застосовуватися до TCP-трафіку, що є одним із основних протоколів

транспортного рівня в моделі OSI. Далі йдуть параметри any any -> any any, що означає, що правило буде застосовуватися до будь-якого джерела (any any) та будь-якого призначення (any any) у мережевому трафіку. Таким чином, правило не обмежується конкретними IP-адресами або портами на початковому етапі визначення потенційної загрози.

Опція msg:"Port Scan Detected (portscanDetection.lua) : Over 9 different ports in 120 sec." задає повідомлення, яке буде відображатися у журналі Suricata при спрацьовуванні алерту. Це повідомлення містить опис загрози, а саме виявлення спроби сканування портів, а також вказує на використання Lua-скрипту portscanDetection.lua для реалізації логіки детекції. Додатково, повідомлення містить інформацію про те, що алерт спрацьовує при виявленні звертань до понад 9 різних портів протягом 120 секунд.

Опція priority:2 визначає пріоритет алерту. Пріоритети в Suricata варіюються від 1 до 3, де 1 є найвищим пріоритетом, що вказує на серйозну загрозу, а 3 — найнижчим, що може свідчити про менш критичні події. В даному випадку, пріоритет 2 вказує на загрозу середньої важливості, яка потребує уваги, але не є критичною для безпеки системи.

Наступна опція flow:to_server визначає напрямок потоку трафіку, який буде аналізуватися правилом. Це означає, що правило буде застосовуватися лише до трафіку, спрямованого до серверів, що допомагає зосередитися на потенційно шкідливих діях, спрямованих до внутрішніх ресурсів мережі.

Опція threshold: type limit, track by_src, seconds 1200, count 1 встановлює межу спрацьовування алерту. Тут визначено, що алерт буде спрацьовувати лише один раз (count 1) на джерело (track by_src) протягом 1200 секунд (20 хвилин). Це допомагає запобігти надмірному генеруванню алертів при постійних або повторюваних спробах сканування портів з однієї і тієї ж IP-адреси, забезпечуючи більш ефективне використання ресурсів для аналізу.

Ключовим елементом цього правила є інтеграція Lua-скрипту, зазначеного в опції lua:portscanDetection.lua. Lua-скрипт відповідає за

розширену логіку детекції, яка не може бути реалізована лише за допомогою стандартних правил Suricata. У цьому випадку, скрипт виконує аналіз трафіку з метою визначення спроб сканування портів, відстежуючи кількість різних портів, до яких звертається певне джерело, і визначає, чи перевищує ця кількість встановлений поріг.

Ідентифікатор правила sid:1000001 є унікальним числом, яке використовується для ідентифікації та управління правилом. Це дозволяє легко відстежувати, оновлювати або видаляти правило при необхідності. Ревізія правила rev:1 вказує на версію правила, що є корисним для відстеження змін та оновлень у правилах безпеки.

Таким чином, даний алерт поєднує стандартні можливості Suricata з розширеною логікою, реалізованою через Lua-скрипт, що дозволяє ефективно виявляти та реагувати на спроби сканування портів у мережевому середовищі.

Скрипт portscanDetection.lua є ключовим компонентом системи виявлення загроз Suricata, який відповідає за розпізнавання спроб сканування портів у мережевому трафіку. Цей скрипт реалізує механізм відстеження активності джерел трафіку та визначення, чи перевищує кількість різних портів, до яких звертається певне джерело, встановлені пороги за певний проміжок часу.

На першому етапі роботи скрипт виконує ініціалізацію необхідних ресурсів. У функції init визначається, що для аналізу потрібні повні мережеві пакети. Це досягається шляхом встановлення залежностей через об’єкт needs, де ключ "packet" активує доступ до вмісту пакетів:

Лістинг 3.2 – Ініціалізація необхідних ресурсів

```
function init(args)
    local needs = {}
    needs["packet"] = tostring(true)
    return needs
end
```

Для забезпечення актуальності даних використовується функція очищення таблиці clear_table, яка видаляє записи про з'єднання після закінчення визначеного інтервалу часу. Це дозволяє уникнути накопичення застарілих даних та забезпечує ефективність аналізу:

Лістинг 3.3 – Видалення застарілих даних

```
function clear_table()
    portscan = {}
end
```

Одним із важливих аспектів роботи скрипта є перетворення рядків часу у формат секунд для подальших розрахунків. У цьому випадку функція time_to_seconds приймає часовий рядок та перетворює його в секунди, використовуючи функцію os.time:

Лістинг 3.4 – Перетворення часу в секунди

```
function time_to_seconds(time_str)
    local year, month, day, hour, min, sec, msec = 
time_str:match("(%d+)/(%d+)/(%d+)-(%d+):(%d+):(%d+).(%d+)")
    return os.time{year=year, month=month, day=day, hour=hour, 
min=min, sec=sec}
end
```

Головна логіка скрипта реалізована у функції match. Вона аналізує кожний пакет, отриманий у мережі, та зберігає інформацію про джерело IP і порти призначення. У разі, якщо кількість унікальних портів для одного джерела перевищує визначений поріг (9 портів), скрипт генерує сигнал сповіщення:

Лістинг 3.5 – Аналіз пакетів, зберігання інформації про з’єднання

```
function match(args)
    local pkt = args["packet"]
    if pkt == nil then
        return 0
    end

    local current_time_str = SCPacketTimeString()
    local current_time = time_to_seconds(current_time_str)
```

Продовження лістингу 3.5

```
    if last_reset_time == nil then
        last_reset_time = current_time
    end

    if os.difftime(current_time, last_reset_time) > 150 then
        clear_table()
        last_reset_time = current_time
    end

    ipver,  src_ip,  dst_ip,  proto,  src_port,  dst_port  =
SCPacketTuple()
    if src_ip == nil or dst_port == nil then
        return 0
    end

    if not portscan[src_ip] then
        portscan[src_ip] = {}
    end

    portscan[src_ip][dst_port] = true

    local port_count = 0
    for port, _ in pairs(portscan[src_ip]) do
        port_count = port_count + 1
    end
    if port_count > 9 then
        return 1
    end
    return 0

end
```

Функція match забезпечує основну функціональність скрипта. Спершу вона перевіряє доступність пакета для аналізу. Далі, використовуючи функцію SCPacketTimeString, отримує часовий рядок пакета, який конвертується у формат секунд за допомогою time_to_seconds. Для кожного джерела IP створюється окремий запис у таблиці portscan, де зберігається інформація про порти призначення. Якщо кількість унікальних портів для одного джерела перевищує визначений поріг, функція повертає значення 1, що свідчить про виявлення портскану.

Під час розробки та тестування Lua-скриптів для виявлення порт-сканування було проведено серію експериментів з використанням утиліти

nmap для моделювання сканування відкритих портів. Це дозволило перевірити, чи правильно працює розроблений Lua-скрипт.

Для тестування було попередньо відкрито ряд портів на хості, щоб продемонструвати можливість виявлення активності. Хоча відкриття портів не є обов'язковим, це спрощує візуалізацію процесу. На рисунку 3.15 наведено список відкритих портів, отриманий за допомогою команди sudo ss -tuln | grep LISTEN (див. рисунок 3.15).

```
root@kostya:/home/kostya# sudo ss -tuln | grep LISTEN
tcp   LISTEN 0      1            0.0.0.0:25         0.0.0.0:*
tcp   LISTEN 0      1            0.0.0.0:25         0.0.0.0:*
tcp   LISTEN 0      1            0.0.0.0:110        0.0.0.0:*
tcp   LISTEN 0      1            0.0.0.0:110        0.0.0.0:*
tcp   LISTEN 0      1            0.0.0.0:135        0.0.0.0:*
tcp   LISTEN 0      1            0.0.0.0:135        0.0.0.0:*
tcp   LISTEN 0      1            0.0.0.0:443        0.0.0.0:*
tcp   LISTEN 0      1            0.0.0.0:443        0.0.0.0:*
tcp   LISTEN 0      1            0.0.0.0:993        0.0.0.0:*
tcp   LISTEN 0      1            0.0.0.0:993        0.0.0.0:*
tcp   LISTEN 0      1            0.0.0.0:995        0.0.0.0:*
```

Рисунок 3.15 – Список відкритих портів

Далі, використовуючи утиліту nmap, ми виконали сканування порту хоста з IP-адресою 192.168.175.128 для виявлення доступних портів. Команда для сканування: nmap -sS -p1-65535 -T4 192.168.175.128

Результат сканування наведено на рисунку 3.16 (див. рисунок 3.16). У ньому показано відкриті порти та відповідні їм сервіси.

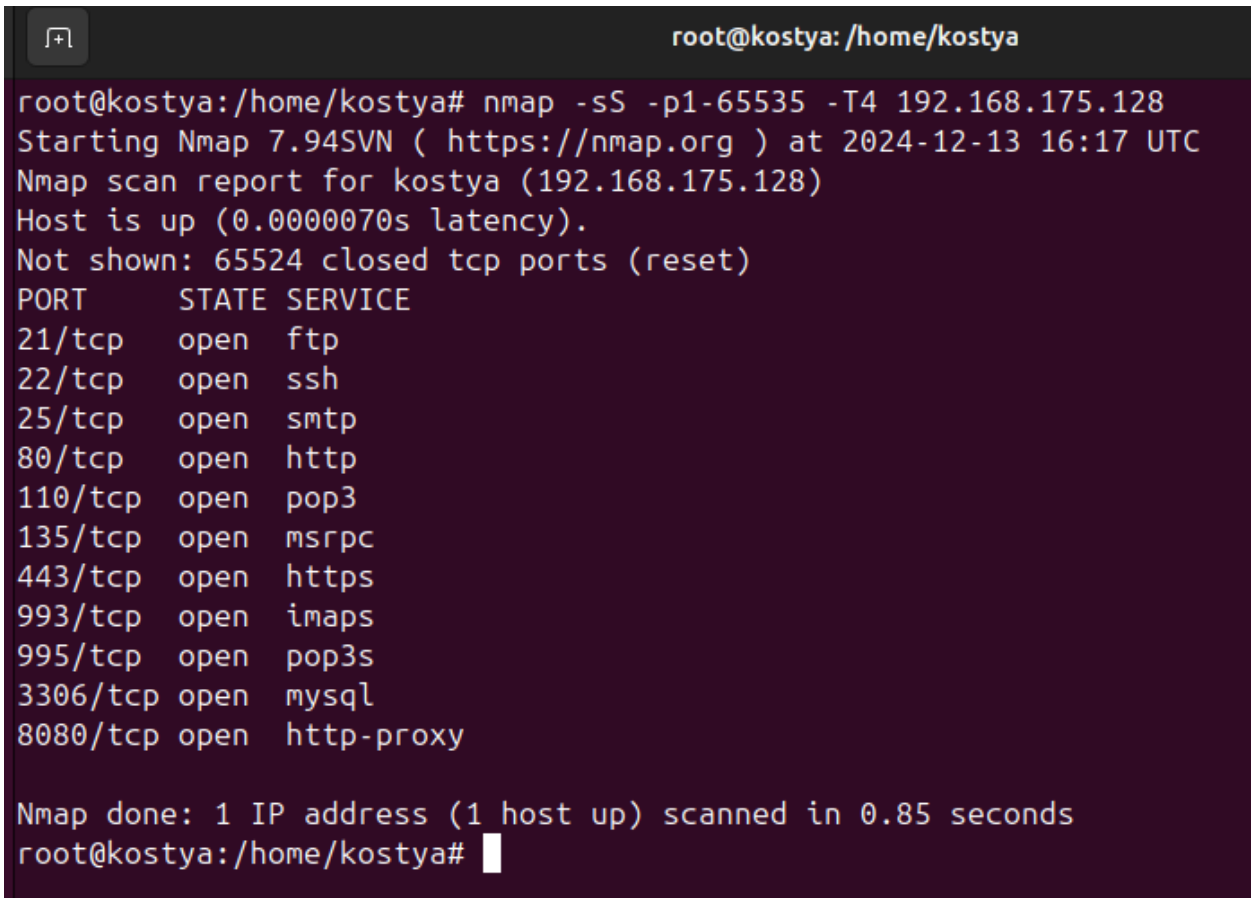

Рисунок 3.16 – Результат сканування портів

Для перевірки роботи Lua-скрипта ми здійснили моніторинг логів Suricata, зокрема файлу eve.json, використовуючи фільтрування через утиліту jq: sudo tail -f /var/log/suricata/eve.json | jq 'select(.event_type=="alert")'

Цей підхід дозволяє оперативно переглядати лише події типу "alert", що включають повідомлення про порт-сканування.

На рисунку 3.17 представлено лог-алерт, створений скриптом, який фіксує сканування більше ніж 9 різних портів у встановлений часовий проміжок (див. рисунок 3.17). Алерт свідчить про виявлення сканування портів на хості з IP-адресою 192.168.175.128. Згідно з інформацією, поданою в алерті, сканування було ініційоване джерелом із IP-адресою 192.168.1.100 на порту 55931, і було спрямоване на порт 80 хоста цілі. Алерт також містить такі поля:

–timestamp – час, коли було зафіксовано подію: 2024-12-13T16:17:04.947275+0000.

–event_type – тип події: alert, що вказує на спрацювання правила.

–alert:

–action – дія, вжита Suricata: allowed, що свідчить про те, що трафік було дозволено.

–signature_id – унікальний ідентифікатор правила: 1000001.

–signature – текстова інформація з правила: "Port Scan Detected (portscanDetection.lua): Over 9 different ports in 120 sec."

–category – категорія події: "Attempted Information Leak".

–severity – рівень серйозності: 2.


```
    "category": "Attempted Information Leak",
    "severity": 2
  }
}
{
  "timestamp": "2024-12-13T16:17:04.947275+0000",
  "flow_id": 1810834761386803,
  "event_type": "alert",
  "src_ip": "192.168.1.100",
  "src_port": 55931,
  "dest_ip": "192.168.175.128",
  "dest_port": 80,
  "proto": "TCP",
  "alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id": 1000001,
    "rev": 1,
    "signature": "Port Scan Detected (portscanDetection.lua) : Over 9 different ports in 12
0 sec.",
    "category": "Attempted Information Leak",
    "severity": 2
  }
}
```


Рисунок 3.17 – Лог-алерт про виявлення порт-сканування

Цей алерт генерується нашим Lua-скриптом, який відстежує кількість унікальних портів, до яких здійснюється підключення з одного джерела IP у межах заданого часового інтервалу (120 секунд). Якщо ця кількість перевищує 9, генерується алерт.

Результати, представлені на рисунку 3.17, демонструють ефективність роботи розробленого Lua-скрипта. Завдяки його алгоритму Suricata змогла виявити спробу порт-сканування, що є одним із ключових індикаторів можливої шкідливої активності в мережі.

Таким чином, ці результати підтверджують коректну роботу Lua-скрипта. Він успішно фіксує спроби сканування портів і генерує відповідний алерт у логах Suricata. Це демонструє, як використання Lua дозволяє розширити функціональні можливості IDS/IPS-систем, забезпечуючи точніше та гнучкіше виявлення мережевих загроз.

РОЗДІЛ 4. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Розробка та впровадження систем виявлення та запобігання вторгненням (IDS/IPS), зокрема з використанням мови Lua для розширення їх функціональних можливостей, вимагає особливої уваги до питань охорони праці. Працівники, які займаються програмуванням, тестуванням та експлуатацією таких систем, проводять значний час за екранними пристроями, що може призвести до різних професійних ризиків. В Україні діють нормативні акти, які регламентують вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями.

Згідно з наказом Міністерства соціальної політики України від 14 лютого 2018 року № 207, затверджено "Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями" [20]. Цей документ встановлює мінімальні вимоги безпеки та захисту здоров'я під час роботи з екранними пристроями незалежно від їх типу та моделі. Він розроблений на основі Директиви 90/270/ЄЕС від 29 травня 1990 року про мінімальні вимоги безпеки та здоров’я при роботі з екранними пристроями.

Роботодавець зобов'язаний інформувати працівників про умови праці та наявність на їх робочих місцях небезпечних та шкідливих виробничих факторів, які виникають під час роботи з екранними пристроями, а також про можливі наслідки їх впливу на здоров'я працівників. Крім того, роботодавець повинен забезпечити навчання і перевірку знань працівників з питань охорони праці та безпечного використання екранних пристроїв до початку роботи з ними, а також у випадках модифікації та організації роботи обладнання [20].

Особлива увага приділяється організації робочого місця. Робочі місця працівників з екранними пристроями мають бути спроектовані так, щоб працівники мали простір для зміни робочого положення та рухів [20].

Освітлення робочого місця повинно створювати відповідний контраст між екраном і навколишнім середовищем та відповідати вимогам Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПіН 3.3.2.007-98 [21].

Мікроклімат виробничих приміщень з робочими місцями працівників з екранними пристроями має підтримуватись на постійному рівні та відповідати вимогам Санітарних норм мікроклімату виробничих приміщень ДСН 3.3.6.042-99 [22]. Робочий стіл або робоча поверхня повинні бути достатнього розміру та мати поверхню з низькою відбивною здатністю, допускати гнучкість під час розміщення екрана, клавіатури, документів і відповідного устаткування.

Робоче крісло має бути стійким і дозволяти працівнику легко рухатися та займати зручне положення. Сидіння повинно регулюватися по висоті, а спинка сидіння — як по висоті, так і по нахилу. Слід передбачати підніжку для тих, кому це необхідно для зручності [20].

Для зменшення навантаження на зір та опорно-руховий апарат рекомендується організовувати внутрішні регламентовані перерви для відпочинку відповідно до ДСанПіН 3.3.2.007-98 [21]. Роботодавець також має забезпечити за свій рахунок проведення медичних оглядів працівників відповідно до вимог Порядку проведення медичних оглядів працівників певних категорій, затвердженого наказом Міністерства охорони здоров'я України від 21 травня 2007 року № 246 [23].

Дотримання цих вимог є критично важливим для забезпечення безпеки та захисту здоров'я працівників. Це сприяє підвищенню продуктивності праці та зменшенню ризику професійних захворювань.

Для забезпечення безпечних умов праці під час роботи з екранними пристроями слід також враховувати особливості використання таких пристроїв у контексті інформаційної безпеки. Працівники, які працюють із системами IDS/IPS, часто стикаються з високою інтенсивністю роботи,

потребою в аналізі великого обсягу даних, що може призводити до стресу та перенапруження. Тому важливим є дотримання не лише технічних, а й організаційних заходів охорони праці.

Одним із ключових аспектів є ергономічність програмного забезпечення, яке використовується для аналізу мережевого трафіку. Згідно з Вимогами щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями, програмне забезпечення має бути простим у використанні, а де необхідно – адаптованим до рівня знань і досвіду працівників [20].

У нашій роботі враховані та виконані всі відповідні норми та вимоги з охорони праці, що забезпечує безпечні та комфортні умови для виконання завдань, пов'язаних із розробкою, тестуванням та впровадженням систем. Робоче місце було облаштоване відповідно до положень ДСанПіН 3.3.2.007-98, які регламентують ергономіку, освітлення та перерви під час роботи з екранними пристроями. Мікроклімат робочого приміщення відповідає ДСН 3.3.6.042-99, що гарантує підтримання оптимальних температурних і санітарно-гігієнічних умов. Крім того, використовуване обладнання повністю відповідає Вимогам щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями, затвердженими наказом Міністерства соціальної політики України.

4.2 Безпека в надзвичайних ситуаціях

Забезпечення безпеки життєдіяльності при роботі з ПК є важливою складовою збереження здоров'я та підвищення продуктивності працівників. Тривала робота за комп'ютером може спричинити низку проблем, зокрема перевтому, порушення зору, опорно-рухового апарату та загальне погіршення стану здоров'я. Для мінімізації негативного впливу необхідно враховувати ергономічні вимоги, забезпечувати безпеку обладнання та створювати комфортні умови праці.

Одним із ключових аспектів є правильна організація робочого місця. Монітор слід розташовувати на рівні очей, щоб уникнути напруження шиї та плечей. Відстань від екрану до очей повинна бути не меншою за 50–70 см, що забезпечить комфортне сприйняття інформації та знизить ризик розвитку короткозорості. Важливим є використання крісла з регульованою висотою та підтримкою попереку, яке забезпечує правильну поставу та знижує навантаження на хребет. Для зменшення статичної напруги необхідно робити короткі перерви для руху та зміни положення тіла [24].

Особливу увагу слід приділяти освітленню робочого місця. Рівномірне, неяскраве освітлення без різких тіней і відблисків на екрані зменшує напруження очей. Крім того, сучасні монітори мають бути обладнані антибліковим покриттям або спеціальними захисними фільтрами. Рекомендується також періодично відволікатися від екрану, фокусуючи погляд на далеких об’єктах, що зменшує напруження очних м’язів.

Мікроклімат приміщення також відіграє важливу роль у забезпеченні безпеки життєдіяльності. Оптимальна температура повітря має бути в межах 18–22 °C, а вологість – 40–60%. Регулярне провітрювання забезпечує приплив свіжого повітря, що позитивно впливає на самопочуття та продуктивність. Варто уникати шуму та зайвих подразників, які можуть відволікати та спричиняти перевтому [24].

Дотримання режиму праці та відпочинку є ще одним важливим аспектом. Для підтримання здоров’я рекомендується робити перерви кожні 50–60 хвилин роботи за комп’ютером. Під час перерв варто виконувати прості фізичні вправи, які покращують кровообіг, знижують напруження м’язів і допомагають зберегти концентрацію.

Небезпека ураження електричним струмом є одним із основних ризиків при роботі з ПК. Для забезпечення електробезпеки важливо використовувати якісні електричні кабелі та розетки, уникати перевантаження мережі та регулярно перевіряти стан обладнання. Правильне розташування

електропроводів і відсутність контакту з вологою допомагають зменшити ризик виникнення аварійних ситуацій.

Довготривала робота за комп'ютером може викликати не лише фізичні, але й психоемоційні проблеми, такі як стрес або професійне вигорання. Для їхньої профілактики необхідно створювати комфортні умови праці, дотримуватись збалансованого графіка роботи та підтримувати позитивну атмосферу на робочому місці. Важливо також знаходити час для активного відпочинку, який сприяє відновленню сил і загальному зміцненню організму [25].

Важливим аспектом здоров'я при роботі з ПК є режим праці та відпочинку. Для збереження продуктивності та запобігання перевтомі рекомендується робити перерви кожні 60 хвилин тривалістю 10–15 хвилин. Під час перерв слід виконувати легкі фізичні вправи для розслаблення м'язів і зниження напруги [26].

Психоемоційний стан людини також зазнає впливу при тривалій роботі з ПК. Постійна напруга може спричинити стрес, який, у свою чергу, негативно впливає на працездатність і здоров'я. Для зменшення стресу рекомендується:

–підтримувати збалансований режим дня, що включає достатній сон і регулярні фізичні вправи;

–чергувати роботу за ПК із іншими видами діяльності;

–облаштувати робоче місце з урахуванням естетичних та психологічних факторів, наприклад, додати елементи декору, які сприяють релаксації [26].

Отже, забезпечення безпеки життєдіяльності при роботі з ПК вимагає комплексного підходу, який включає ергономіку робочого місця, дотримання режиму праці та відпочинку, контроль стану обладнання та створення комфортного мікроклімату. Виконання цих заходів дозволяє зберегти здоров'я, підвищити продуктивність і уникнути негативних наслідків тривалої роботи за комп'ютером.

ВИСНОВКИ

У магістерській роботі досліджено можливості використання мови Lua для розширення функціональних можливостей систем виявлення та запобігання вторгнень (IDS/IPS) на прикладі платформи Suricata. Проведений аналіз сучасних підходів до забезпечення кібербезпеки показав, що традиційні методи виявлення загроз є недостатньо ефективними для протидії сучасним атакам, таким як порт-сканування чи використання складних обхідних технік. Використання мови Lua дозволяє створювати адаптивні рішення, які підвищують гнучкість та ефективність роботи IDS/IPS.

У процесі дослідження здійснено детальний аналіз механізмів інтеграції Lua у Suricata. Розглянуто переваги використання Lua, серед яких: простота синтаксису, висока продуктивність, гнучкість у створенні користувацьких правил і можливість інтеграції з іншими системами кіберзахисту, такими як Threat Intelligence та SIEM. На основі проведеного аналізу розроблено та протестовано Lua-скрипти, які демонструють здатність виявляти мережеві загрози, зокрема спроби порт-сканування, та виконувати гнучке логування HTTP-трафіку.

Результати тестування підтвердили ефективність розроблених рішень. У реальному середовищі вдалося успішно виявити спроби порт-сканування за допомогою спеціально розробленого Lua-скрипта. Усі розроблені рішення показали стабільну роботу та можливість інтеграції у платформу Suricata без змін у її основному коді.

Наукова новизна роботи полягає у впровадженні Lua-скриптів для створення гнучких правил виявлення та аналізу загроз. Зокрема, вперше запропоновано реалізацію виявлення порт-сканування на основі Lua для Suricata з використанням нових алгоритмів обробки трафіку. Також показано можливість інтеграції зовнішніх джерел даних у процес аналізу трафіку, що відкриває перспективи для створення більш складних і точних систем кіберзахисту.

Практичне значення роботи полягає у створенні прикладного рішення, яке може бути впроваджене в корпоративні мережі, державні установи та об'єкти критичної інфраструктури для забезпечення високого рівня кібербезпеки. Використання мови Lua дозволяє швидко адаптувати систему Suricata до нових викликів та створювати користувацькі правила для специфічних сценаріїв.

Отримані результати можуть бути використані для подальшого вдосконалення систем IDS/IPS, зокрема для інтеграції з іншими платформами кіберзахисту, такими як SOAR. Розроблені методи та скрипти відкривають перспективи для побудови адаптивних і високоефективних рішень у сфері інформаційної безпеки.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is an intrusion detection and prevention system (IDPS)? (б.д.). Red Hat - We make open source technologies for the enterprise. https://www.redhat.com/en/topics/security/what-is-an-IDPS

2. Tymoshchuk, V., Pakhoda, V., Dolinskyi, A., Karnaukhov, A., & Tymoshchuk, D. (2024). MODELLING CYBER THREATS AND EVALUATING THE PERFORMANCE OF INTRUSION DETECTION SYSTEMS. Grail of Science, (46), 636–641. https://doi.org/10.36074/grail-of-science.29.11.2024.081

3. Tymoshchuk, V., Mykhailovskyi, O., Dolinskyi, A., Orlovska, A., & Tymoshchuk, D. (2024). OPTIMISING IPS RULES FOR EFFECTIVE DETECTION OF MULTI-VECTOR DDOS ATTACKS. Матеріали конференцій МЦНД, (22.11. 2024; Біла Церква, Україна), 295-300.

4. Yakymenko, I., Karpinski, M., Shevchuk, R., & Kasianchuk, M. (2024). Symmetric Encryption Algorithms in a Polynomial Residue Number System. Journal of Applied Mathematics, art. id 4894415, 1-12.

5. Tymoshchuk, D., Yasniy, O., Mytnyk, M., Zagorodna, N., Tymoshchuk, V., (2024). Detection and classification of DDoS flooding attacks by machine learning methods. CEUR Workshop Proceedings, 3842, pp. 184 - 195.

6. Al-Yaseen, W. L., Othman, Z. A., & Nazri, M. Z. A. (2017). A multi-level hybrid model for intrusion detection systems. Expert Systems with Applications, 72, 1–15.

7. Kuznetsov, O., Poluyanenko, N., Frontoni, E., Kandiy, S., Karpinski, M., & Shevchuk, R. (2024). Enhancing Cryptographic Primitives through Dynamic Cost Function Optimization in Heuristic Search. Electronics, 13(10), 1-52.

8. Lypa, B., Horyn, I., Zagorodna, N., Tymoshchuk, D., Lechachenko T., (2024). Comparison of feature extraction tools for network traffic data. CEUR Workshop Proceedings, 3896, pp. 1-11.

9. Tymoshchuk, V., Vantsa, V., Karnaukhov, A., Orlovska, A., & Tymoshchuk, D. (2024). COMPARATIVE ANALYSIS OF INTRUSION DETECTION APPROACHES BASED ON SIGNATURES AND ANOMALIES. Матеріали конференцій МЦНД, (29.11. 2024; Житомир, Україна), 328-332.

10. Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2018). A deep learning approach to network intrusion detection. IEEE Transactions on Emerging Topics in Computational Intelligence, 2(1), 41–50.

11. Yesin, V., Karpinski, M., Yesina, M., Vilihura, V., Kozak, R., & Shevchuk, R. (2023). Technique for Searching Data in a Cryptographically Protected SQL Database. Applied Sciences, 13(20), art. no. 11525, 1-21.

12. Yin, C., Zhu, Y., Fei, J., & He, X. (2017). A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access, 5, 21954–21961.

13. Tymoshchuk, V., Vorona, M., Dolinskyi, A., Shymanska, V., & Tymoshchuk, D. (2024). SECURITY ONION PLATFORM AS A TOOL FOR DETECTING AND ANALYSING CYBER THREATS. Collection of scientific papers «ΛΌГОΣ», (December 13, 2024; Zurich, Switzerland), 232-237.

14. Ierusalimschy, R. (2013). Programming in Lua (3rd ed.). Lua.org. Retrieved from https://www.lua.org/pil/3.1.html

15. Lua: The Lightweight Scripting Language Powering Cybersecurity Tools and Threat Analysis. (2024, 30 жовтня). isecjobs.com. Retrieved from https://isecjobs.com/insights/lua-explained/

16. Snort IDS/IPS Explained: What - Why you need - How it works. (n.d.). Zenarmor. Retrieved from https://www.zenarmor.com/docs/network-security-tutorials/what-is-snort

17. ТИМОЩУК, Д., & ЯЦКІВ, В. (2024). USING HYPERVISORS TO CREATE A CYBER POLYGON. MEASURING AND COMPUTING DEVICES IN TECHNOLOGICAL PROCESSES, (3), 52-56.

19. ТИМОЩУК, Д., ЯЦКІВ, В., ТИМОЩУК, В., & ЯЦКІВ, Н. (2024). INTERACTIVE CYBERSECURITY TRAINING SYSTEM BASED ON SIMULATION ENVIRONMENTS. MEASURING AND COMPUTING DEVICES IN TECHNOLOGICAL PROCESSES, (4), 215-220.

20. (б. д.). Enterprise Open Source and Linux | Ubuntu. https://ubuntu.com/download/desktop

21. Tymoshchuk, D., & Yatskiv, V. (2024). Slowloris ddos detection and prevention in real-time. Collection of scientific papers «ΛΌГОΣ», (August 16, 2024; Oxford, UK), 171-176.

22. Про затвердження Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин. (б. д.). Офіційний вебпортал парламенту України. https://zakon.rada.gov.ua/rada/show/v0007282-98#Text

23. Про затвердження Санітарних норм мікроклімату виробничих приміщень. (б. д.). Офіційний вебпортал парламенту України. https://zakon.rada.gov.ua/rada/show/va037282-99#Text

24. Про затвердження Порядку проведення медичних оглядів працівників певних категорій. (б. д.). Офіційний вебпортал парламенту України. https://zakon.rada.gov.ua/laws/show/z0846-07#Text

25. Яскілка, В. Я., & Олійник, М. З. (2016). Конспект лекцій з курсу «Охорона праці в галузі». Тернопіль: Вид-во ТНТУ імені Івана Пулюя.

26. Бадищук, В. І., & Чихіра, І. В. (2016). Охорона праці в галузі. Конспект лекцій. Тернопіль: Тернопільський національний технічний університет ім. Івана Пулюя.

27. Кафедра охорони праці, промислової та цивільної безпеки. (2020, 11 лютого). Охорона праці та ПК, як безпечно працювати на персональному комп’ютері. Кафедра охорони праці, промислової та цивільної безпеки. Retrieved from https://opcb.kpi.ua/?p=3590

ДОДАТОК А ПУБЛІКАЦІЯ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
імені ІВАНА ПУЛЮЯ
НАУКОВЕ ТОВАРИСТВО ім. ШЕВЧЕНКА

М А Т Е Р І А Л И

XII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

18-19 грудня 2024 року

ТЕРНОПІЛЬ
2024

УДК 001
М34

ПРОГРАМНИЙ КОМІТЕТ

Голова: Приймак Микола – професор кафедри комп'ютерних систем та мереж, д.т.н., професо**р.**

Співголови: Марущак Павло – проректор з наукової роботи, докт. техн. наук, професор.

Баран Ігор – канд. техн. наук, доцент, декан факультету ФІС.

Науковий секретар: Надія Крива– старший викладач.

Члени: Василь Кривень - завідувач кафедри математичних методів в інженерії д.ф.-м.н., професор; Галина Осухівська – завідувач кафедри комп'ютерних систем та мереж, к.т.н., доцент; Микола Карпінський - професор кафедри кібербезпеки, д.т.н., професор; Жанна Баб'як - завідувач кафедри української та іноземних мов, к.пед. н., доцент; Ярослав Литвиненко – професор кафедри комп'ютерних наук, д.т.н., професор; Михайло Петрик - завідувач кафедри програмної іженерії, д.ф.-м.н., професор; Наталія Загородна – завідувач кафедри кібербезпеки, к.т.н., доцент.

ОРГАНІЗАЦІЙНИЙ КОМІТЕТ

Голова: Скоренький Юрій Любомирович – канд. техн. наук, доцент кафедри фізики.

Члени: доцент кафедри комп'ютерних наук, к.т.н. В. Никитюк; доцент кафедри програмної інженерії, к.т.н. Д. Михалик; доцент кафедри кібербезпеки, к.т.н. М. Стадник; доцент кафедри комп'ютерних систем та мереж, к.т.н. Є. Тиш; ст. викладач Л. Джиджора.

М34 Матеріали XII науково-технічної конфції «Інформаційні моделі, системи та технології» Тернопільського національного технічного університету імені Івана Пулюя, (Тернопіль, 18–19 грудня 2024 р.). – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2024.

Адреса оргкомітету: ТНТУ ім. І. Пулюя, м. Тернопіль, вул. Руська, 56, 46001,
тел. (0352) 52-41-33, факс (0352) 25-49-83.
E-mail: conffis2024@gmail.com

Редагування, оформлення та верстка: Надія Крива

СЕКЦІЇ КОНФЕРЕНЦІЇ, ЯКІ ПРЕДСТВЛЕНІ В ЗБІРНИКУ

- Математичне моделювання
- Інформаційні системи та технології, кібербезпека
- Комп'ютерні системи та мережі
- Програмна інженерія та моделювання складних розподілених систем
- Новітні фізико-технічні та освітні технології

В збірнику надруковано тези доповідей XII науково-технічної конференції «Інформаційні моделі, системи та технології» (Тернопіль, 18–19 грудня 2024 р.) за такими науковими напрямками: математичне моделювання; інформаційні системи та технології, кібербезпека; комп'ютерні системи та мережі; програмна інженерія та моделювання складних розподілених систем; новітні фізико-технічні та освітні технології.

Розрахований на науковців, викладачів та студентів вузів.

За зміст тез та дотримання норм академічної доброчесності відповідальність несе автор.

УДК 004.77
Чурбаков Константин, студент гр. СБм-62
Тернопільський національний технічний університет імені Івана Пулюя

ВИКОРИСТАННЯ МОВИ LUA ДЛЯ РОЗШИРЕННЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ IDS/IPS

Churbakov Konstantyn, student of gr. СБм-62

USING LUA LANGUAGE TO EXTEND THE FUNCTIONALITY OF IDS/IPS

Інформаційна безпека є важливим напрямком у сучасних технологіях, а системи виявлення та попередження вторгнень (IDS/IPS) відіграють ключову роль у захисті мереж. Однак стандартні функціональні можливості таких систем можуть бути недостатніми для специфічних потреб, що створює попит на інструменти їхньої адаптації. Використання мови Lua дозволяє створювати гнучкі налаштування та розширювати функціонал IDS/IPS, забезпечуючи адаптивність до нових загроз та підвищення ефективності їх роботи. [1].

Впровадження Lua в системи виявлення вторгнень дозволяє ефективно інтегрувати специфічні сценарії аналізу мережевого трафіку, що важливо для виявлення складних і невідомих типів атак. Можливість використання користувацьких скриптів розширює функціональність базових механізмів IDS/IPS, роблячи систему більш гнучкою до змін у кіберзагрозах [2].

Метою дослідження є розробка і тестування інтеграції мови Lua з існуючими IDS/IPS для створення динамічних модулів, здатних забезпечити більш точне виявлення загроз і зменшення кількості хибних спрацьовувань. У роботі застосовувалася мова Lua для створення користувацьких скриптів, які інтегруються із платформою Suricata, що забезпечує обробку мережевого трафіку з дотриманням специфічних правил. Локальне тестування показало підвищення точності і надійності системи виявлення. [3].

Особливістю мови Lua є її легкість і висока продуктивність, що робить її ідеальним вибором для інтеграції у реальному часі в умовах великих обсягів мережевого трафіку. Lua дозволяє створювати правила та логіку, які працюють паралельно з основними алгоритмами аналізу без зниження загальної продуктивності системи.

Результати роботи демонструють, що впровадження мови Lua у роботу IDS/IPS може суттєво розширити функціональні можливості цих систем, зокрема шляхом динамічного налаштування правил та адаптації до сучасних викликів у сфері інформаційної безпеки.

Література:

1. Paxson V. Bro: A System for Detecting Network Intruders in Real-Time. URL: https://www.usenix.org/legacy/events/sec99/full_papers/paxson/paxson.pdf
2. Roesch M. Snort: Lightweight Intrusion Detection for Networks. URL: https://www.snort.org/downloads/snort/white_paper.pdf
3. Schlueter K. Lua Programming for Real-World Security. O'Reilly Media, 2020.

Додаток Б Лістинг файлу portscanDetection.lua

```
-- Таблиця для відстеження джерел IP-адрес і портів
призначення
local portscan = {}
local last_reset_time = nil

-- Ініціалізація: визначення необхідних ресурсів для роботи
скрипта
function init(args)
    local needs = {}
    needs["packet"] = tostring(true) -- Використовуємо
повні пакети для аналізу
    return needs
end

-- Функція для очищення таблиці portscan через певний
інтервал часу
function clear_table()
    portscan = {}
end

-- Функція для конвертації рядка часу у секунди
function time_to_seconds(time_str)
    local year, month, day, hour, min, sec, msec =
time_str:match("(%d+)/(%d+)/(%d+)-(%d+):(%d+):(%d+).(%d+)")
    return os.time{year=year, month=month, day=day,
hour=hour, min=min, sec=sec}
end

-- Основна логіка аналізу трафіку
function match(args)
    local pkt = args["packet"]

    -- Перевірка, чи передано пакет
    if pkt == nil then
        return 0
    end

    -- Отримуємо час поточного пакета
    local current_time_str = SCPacketTimeString()
    local current_time = time_to_seconds(current_time_str)

    -- Ініціалізація часу останнього скидання таблиці, якщо
це перше викликання
    if last_reset_time == nil then
        last_reset_time = current_time
    end
```

Продовження лістингу додатку Б

```
    -- Очищаємо таблицю, якщо минуло більше 150 секунд від
останнього скидання
    if os.difftime(current_time, last_reset_time) > 150
then
        clear_table()
        last_reset_time = current_time
    end

    -- Витягуємо IP-адресу джерела та порт призначення з
пакета
    ipver, src_ip, dst_ip, proto, src_port, dst_port =
SCPacketTuple()

    -- Перевіряємо, чи є IP-адреса джерела та порт
призначення
    if src_ip == nil or dst_port == nil then
        return 0
    end

    -- Якщо IP-адреса ще не в таблиці, додаємо її
    if not portscan[src_ip] then
        portscan[src_ip] = {}
    end

    -- Додаємо порт призначення до списку для цієї IP-
адреси
    portscan[src_ip][dst_port] = true

    -- Рахуємо кількість унікальних портів призначення
    local port_count = 0
    for port, _ in pairs(portscan[src_ip]) do
        port_count = port_count + 1
    end

    -- Генеруємо сповіщення, якщо кількість унікальних
портів перевищує 9
    if port_count > 9 then
        return 1
    end

    return 0
end
```