МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

РУДИЙ МАКСИМ ВІКТОРОВИЧ

Допускається до захисту:
Завідувач кафедри
інформаційних технологій,
к.т.н., доцент,
_____Нескородєва Т. В.
«__»_____20__р.

РОЗРОБКА ВІРТУАЛЬНОГО СТЕНДУ ДЛЯ АНАЛІЗУ СИСТЕМ ВИЯВЛЕННЯ ВТОРГНЕНЬ

Спеціальність 125 Кібербезпека

Кваліфікаційна (бакалаврська) робота

Науковий керівник:

Загоруйко Л.В.,
к.т.н., доцент кафедри
інформаційних технологій

_____
(підпис)

Оцінка : _____ / _____ / _____
(бали/за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця 2021

АНОТАЦІЯ

Рудий М. В. Розробка віртуального стенду для аналізу систем. Спеціальність 125 «Кібербезпека». Донецький національний університет імені Василя Стуса, 2021.

У кваліфікаційній роботі проводиться аналіз систем виявлення вторгнень, розробка віртуального стенду для тестування працездатності систем виявлення вторгнень і інструкції по його налаштуванню та експлуатації. Перевірка систем виявлення вторгнень на базі віртуального стенду.

Ключові слова: інформаційна безпека, мережеві атаки, система виявлення вторгнень, Suricata, Snort, VirtualBox, Ubuntu, Kali Linux.

Табл. 4. Рис. 18. Бібліограф.: 19 найм.

Rudy M. Development of a virtual stand for systems analysis. Specialty 125 "Cybersecurity". Vasyl Stus Donetsk National University, 2021.

The qualification work analyzes the intrusion detection systems, the development of a virtual stand for testing the performance of intrusion detection systems and instructions for its setup and operation. Verification of intrusion detection systems based on a virtual stand.

Keywords: information security, network attacks, intrusion detection system, Suricata, Snort, VirtualBox, Ubuntu, Kali Linux.

Table. 4. Fig. 18. Bibliography: 19 items.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

СВВ - системи виявлення вторгнень;

IDS - intrusion Detection System;

IPS – система протидії вторгнень;

ІБ – інформаційна система;

NIDS – система мережевого виявлення вторгнень;

ПЗ – програмне забезпечення;

ПАК – програмно апаратний комплекс;

ОС – операційна система;

OSI – модель взаємодії відкритих систем;

DoS – відмова в обслуговуванні;

SQL – мова структурованих запитів;

ICMP – протокол міжмережевих керуючих повідомлень;

IP – міжмережевий протокол;

БД – база даних.

ВСТУП

На сьогоднішній день стрімкий розвиток інформаційних технологій і мережі інтернет є невід'ємною частиною розвитку сучасного суспільства. Однак, з розвитком технологій, зростає кількість проблем, пов'язаних з інформаційною безпекою (ІБ) і її забезпеченням, а також способи реалізації ІБ.

Основна проблема експлуатації систем виявлення вторгнень в освітніх цілях полягає в недоцільності побудови на реальному макеті, з огляду на використання великої кількості обчислювальних засобів техніки та мережевого обладнання, необхідних для побудови, а також дорожнечу обслуговування.

Використання віртуального стенда дозволяє скоротити кількість обладнання до одного комп'ютера з встановленим програмним забезпеченням віртуалізації.

Дання робота направлена на огляд таких проблем як:

1) необхідність аналізу ринку існуючих систем виявлення вторгнень (СВВ);
2) отримання практичних навичок в роботі з системою виявлення вторгнення на віртуальному стенді.

Метою цього дипломного проекту є розробка віртуального стенда для тестування працездатності систем виявлення вторгнень (СВВ).

Практичне застосування полягає в перспективному використанні розробленого віртуального стенда в навчальному процесі підготовки фахівців з ІБ.

Для досягнення поставлених цілей необхідно вирішити наступні завдання:

1) проаналізувати ринок існуючих систем виявлення вторгнень (СВВ);
2) підібрати необхідне програмне забезпечення для розгортання віртуального стенду;
3) вибрати програму віртуалізації віртуального стенду;
4) підібрати операційні системи для віртуальних машин;
5) вибрати систем виявлення вторгнень (СВВ) які будуть встановленні;

6) повністю встановити і налаштувати системи виявлення вторгнень (СВВ) на віртуальному стенді, перевірити їх працездатність.

РОЗДІЛ 1. АНАЛІЗ СИСТЕМ ВИЯВЛЕННЯ ТА ПРОТИДІЇ ВТОРГНЕНЬ

1.1. Аналіз систем виявлення вторгнень

Системи виявлення вторгнень (*Intrusion Detection System*) - це сукупність програмних і / або апаратних засобів, які служать для виявлення фактів несанкціонованого доступу в комп'ютер або комп'ютерну мережу, а також запобігання несанкціонованого управління ними.

Системи виявлення вторгнень використовуються для виявлення деяких типів шкідливої активності, яка може негативно вплинути на безпечність комп'ютерної системи. До такої активності відносяться мережеві атаки проти вразливих сервісів, атаки, спрямовані на підвищення привілеїв, неавторизований доступ до важливих файлів, а також дії шкідливого програмного забезпечення (комп'ютерних вірусів, троянів і черв'яків)

Системи *IDS* дозволяють автоматизувати процес моніторингу та аналізу подій, які відбуваються в мережі або окремо взятому вузлі (системі) з метою виявлення атаки або проникнення. Зазвичай система *IDS* включає в себе три стандартних модуля: модуль інформаційних джерел, модуль аналізу і модуль відповіді. Система буде отримувати інформацію про подію від одного або декількох інформаторів, виконувати визначенні налаштуваннями і конфігурацією аналіз даних цієї події, а потім створювати спеціальні відповіді. В якості відповідей можуть бути як найпростіші звіти, так і активний захист об'єкту при виявленні спроби проникнення. [1, 3]

На відміну від системи виявлення вторгнення (СВВ) антивірус є останнім лінією оборони комп'ютерної безпеки. І якщо СВВ працює на рівні мережі, то антивірус встановлюється і працює на конкретному комп'ютері. Тобто якщо на комп'ютері, який знаходиться в мережі, що захищається СВВ, спрацює антивірус, це означає, що СВВ упустила шкідливі програми та атаку.

Фаєрволи є відмінним механізмом створення бар'єру, що дозволяє перегороджувати вхід певним типам мережевого трафіку і пропускати інші.

Хоча і фаєрвол і *IDS* відносяться до комплексу мережевої безпеки, вони працюють по-різному. Фаєрволл спрямований «назовні», і призначений для запобігання зовнішніх загроз. *IDS* визначає «зсередини» можливе вторгнення і сигналізує про це. Існують також і активні системи, які при визначенні зовнішнього вторгнення відправляють команду міжмережевому екрану блокувати дане з'єднання. Також *IDS* здатні визначати загрози, що виникли зсередини мережі, поза сферою дії брандмауера.

Дії, які під час моніторингу виробляють *IDS*, зазвичай ділять на три групи:

- запис інформації щодо подій: зазвичай інформація зберігається локально, але може бути відправлена в будь-яку централізовану систему збору лог-файлів або *SIEM*-систему
- повідомлення адміністраторів безпеки про інциденти ІБ. Такий вид повідомлення називається *alert* і може здійснюватися по декількох каналах: e-mail, *SNMP*-трапи, повідомлення системного журналу, консоль управління системи IDS. Можлива також програмована реакція з використанням скриптів;
- генерація звітів. Звіти створюються з метою підсумовування всієї інформації по запитуваній події (подіям).

1.2. Аналіз систем протидії вторгнень

Різниця між *IDS* та *IPS* в тому що *IDS* це система, яка відстежує мережу і виявляє невідповідні, неправильні або аномальні дії, в той час як *IPS* це система, яка виявляє вторгнення або атаку і вживає активних заходів для їх запобігання. Основна відмінність між ними полягає в тому, що на відміну від *IDS, IPS* активно вживає заходів для запобігання або блокування виявлених вторгнень. Ці заходи запобігання включають такі дії, як відкидання шкідливих пакетів і скидання або блокування трафіку, що надходить з шкідливих *IP*-адрес. *IPS* можна розглядати як розширення *IDS*, яке має додаткові можливості для запобігання вторгнень при їх виявленні. [2, 7]

Крім очевидних плюсів дані системи мають свої мінуси. Як приклад, *IPS* не завжди має можливість з достатньою точністю визначити інцидент ІБ, або помилково прийняти за інцидент поведінку трафіку в межах норми. Такі випадки прийнято називати *false negative*, у другому випадку говорять *false positive*. [2]

Слід мати на увазі, що неможливо повністю виключити виникнення інцидентів, тому організація в кожному випадку повинна самостійно визначити: ризики який з двох груп слід або мінімізувати, або прийняти як належне.

Існують різні методики виявлення інцидентів за допомогою технологій *IPS*. Більшість реалізацій *IPS* використовують суму даних технологій для того, щоб забезпечити більш високу ступінь детектування загроз. [2]

1.3. Аналіз існуючою класифікації систем виявлення вторгнень

Способи моніторингу системи.

1. Мережеві системи виявлення вторгнень (*NIDS*, від англ. *Network intrusion detection system*) - аналізують мережевий трафік за даними сенсорів, розташованих у ключових вузлах мережі;
2. Системи виявлення вторгнень на рівні хоста (*HIDS*, від англ. *Host-based intrusion detection system*) - виявляють вторгнення за допомогою спеціальної служби, яка аналізує системні запити, лог-файли активності додатків, зміни файлової системи і інші процеси, що відбуваються на рівні хоста;
3. Заснована на прикладних протоколах СВВ (*Application Protocol-based IDS, APIDS*) - це система (або агент), яка веде спостереження і аналіз даних, що передаються з використанням специфічних для певних програм протоколів. Наприклад, на веб-сервері з *SQL* базою даних СВВ буде відстежувати вміст *SQL* команд, що передаються на сервер;

4. Протокол-Орієнтована система виявлення вторгнень (СВВ) (*Protocol-Based intrusion detection system, PIDS*) відстежує і аналізує комунікаційні протоколи зі зв'язаними системами або користувачами, тобто аналізує трафік певних протоколів. Найчастіше встановлюється на веб-сервер для аналізу *HTTP* трафіку. У такому випадку для аналізу шифрованого *HTTPS* трафіку необхідно переглядати пакети до їх шифрування;
5. Гібридна систем виявлення вторгнень (СВВ) (*Hybrid intrusion detection system, Hybrid IDS*) поєднує два і більше підходів до розробки СВВ. Дані від різних сенсорів на різних хостах комбінуються з мережевою інформацією для створення найбільш повного уявлення про безпеку мережі.

Способи аналізу даних.

Після збору інформації вона аналізується, відповідно до цього критерію, СВВ може працювати в двох режимах: аналіз у реальному часі або відкладений аналіз:

1. Аналіз у реальному часі вимагає ресурсів і вимагає великих обчислювальних потужностей. Під час роботи, системі потрібно перевіряти вражаючий обсяг інформації в режимі реального часу. При масштабуванні мережі та збільшенні пропускної здатності також необхідно збільшити ресурси, виділені для роботи СВВ, оскільки як тільки рівень трафіку перевищує експлуатаційні межі, пакети скидаються, і система стає неефективною; [5]
2. Відкладений аналіз, на відміну від аналізу в режимі реального часу, проводить аналіз лише в певний час. Приблизно раз на десять хвилин, раз на годину, раз на день або на прохання адміністратора безпеки. Це рішення значно зменшує навантаження на центральний процесор машини і дозволяє решті програм нормально функціонувати, крім відсутності ситуації падіння пакетів, оскільки всі пакети вже записані та ймовірність пропустити атаку зменшується. [5]

Способи реалізації СВВ.

Існує два способи реалізації системи: програмний та апаратний.

1. Програмна реалізація - це програмний продукт, який встановлюється в операційній системі, такі як *Windows, Linux* та інших *UNIX*-подібних системах. У цьому випадку організація вибирає робочу станцію, найчастіше сервер, на якому встановлена система. Такий підхід мінімізує або виключає витрати на придбання додаткового спеціального обладнання. Крім того, серед програмного забезпечення СВВ є безкоштовні проекти. Звичайно, потрібно розуміти, що користуючись безкоштовними системами, неможна отримати таку ж хорошу підтримку, частоту оновлення та допомогу з усуненням несправностей, як із платними варіантами. Але його використання дозволить зрозуміти принцип роботи, недоліки та переваги, що надалі дасть уявлення про те, що знадобиться від платних систем; [1]
2. Апаратна реалізація- це готовий до використання фізичний пристрій, на якому встановлена операційна система, як правило, дистрибутив *Linux* із встановленим програмним продуктом. Такі розподіли, як правило, значно зменшуються, залишаючи лише утиліти та пакети, необхідні для підвищення продуктивності. Перевага цього методу полягає в тому, що постачальник, який постачає товар, повинен забезпечити, щоб на цьому пристрої його продукт працював із заявленими характеристиками, тобто він буде працювати без перебоїв. [1]

Технологія аналізу СВВ.

1. Із збереженням стану - інформація про минулі події зберігається та враховується, коли система виявлення вторгнень приймає рішення;
2. Без збереження стану - кожна подія розглядається незалежно від інших.

Архітектура СВВ.

1. Централізовані – всі обрахування відбуваються на одній робочій станції;
2. Розподілені системи - це системи, що складаються з декількох елементів. Це можуть бути датчики, розподілені по мережі та різні консолі адміністратора.

Основні компоненти архітектури СВВ.

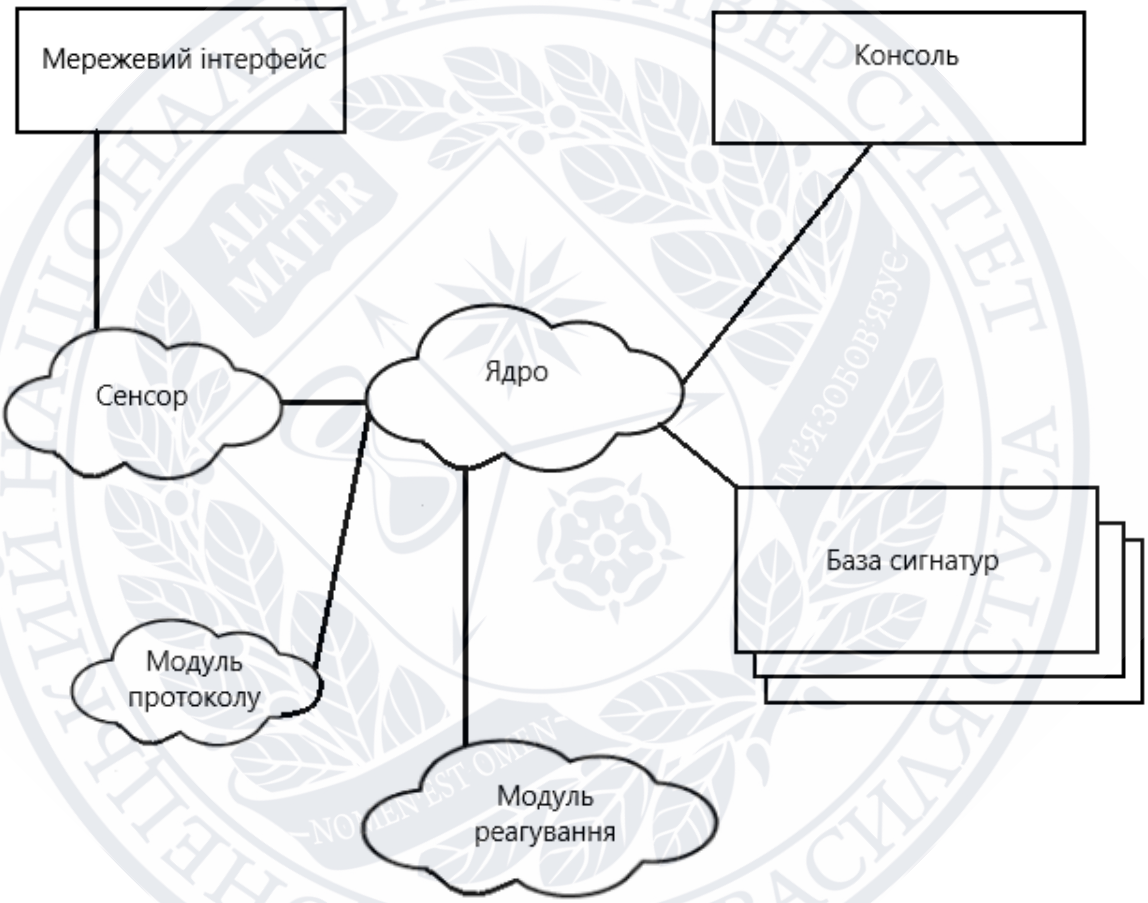

Рисунок 1.3.1 – Архітектура СВВ

1. Датчики - датчики використовуються для моніторингу подій, пов’язаних із безпекою захищеної системи;

2. Аналізатори: аналізують підозрілу активність на основі інформації, отриманої від датчиків. Потім вони формують звіти з результатами аналізу та керують процесами реагування на виявлені випадки;
3. Сховище - забезпечує збір даних, про події і аналіз результатів;
4. Консоль управління - з її допомогою оператор налаштовує СВВ, аналізує інциденти та контролює стан захищеної системи.

1.4. Аналіз методів виявлення вторгнень

Сигнатурний метод.

Сигнатурою прийнято називати шаблон, який відповідає відповідній атаці. Виявлення атаки на основі конкретних підписів - це процес порівняння шаблону підпису з можливим інцидентом.

Приклади сигнатур: [6]

- підключення *telnet* "*root*" користувача, що порушує певні політики безпеки компанії;
- вхідний електронний лист із темою "безкоштовні книги" та вкладений файл "*freebook.exe*";
- Журнал операційної системи з кодом 645, що означає, що аудит хоста вимкнений.

Данна методика ефективна при виявленні уже відомих загроз, але дуже не ефективна коли це нова загроза.

Аномальний метод.

Метод заснований на порівняльному аналізі активності подій з можливими відхиленнями від нормального (зазначеного) рівня. В СВВ "*IPS*", що використовує цей метод, включені так звані профілі, які визначають нормальну поведінку користувачів, з'єднань, вузлів мережі, трафіку та додатків. Ці профілі зазвичай створюються під час так званого "періоду навчання", тривалість якого залежить від обраних алгоритмів розрахунку.

Наприклад, у профілі можна спостерігати збільшення веб-трафіку на 20-30% протягом тижня. Згодом система *IPS* використовуватиме статистичні методи для порівняння різних характеристик діяльності в реальному часі із заздалегідь визначеним пороговим значенням. Якщо це значення перевищено, відповідне повідомлення буде надіслано на консоль управління адміністратора безпеки. Профілі можна створювати на основі різноманітних атрибутів, взятих з аналізу поведінки користувачів. Наприклад, статистика переданих електронних листів, рівень використання процесора сервера за певний проміжок часу, невдалі спроби входу в систему тощо. Як результат, цей метод дозволяє з високою ймовірністю відбивати атаки, які змогли обійти фільтрацію аналізу сигнатур, забезпечуючи таким чином оптимальний рівень захисту від шкідливих атак.

1.5. Аналіз системи мережевого виявлення вторгнень

Мережева система виявлення вторгнень (англ. Network intrusion detection system, *NIDS*) - система виявлення вторгнень, яка відстежує такі види шкідливої діяльності, як *DoS* атаки, сканування портів або навіть спроби проникнення в мережу.

Мережева СВВ переглядає всі вхідні пакети на наявність в них підозрілих ознак. Якщо, наприклад, виявлена велика кількість запитів на *TCP* з'єднання з широким діапазоном різних портів, то, скоріш за все, проводиться сканування портів. Також подібна система найчастіше відстежує вхідний шелл-код схожим чином зі звичайною СВВ.

Мережева СВВ не обмежується відстеженням тільки входить мережевого трафіку. Часто важливу інформацію про те, що відбувається вторгнення можна отримати також з вихідного або локального трафіку. Дія деяких атак може розгортатися всередині спостерігається мережі або сегмента мережі, і ніяк не відбиватися на вхідному трафіку. [7]

Найчастіше, мережева СВВ добре взаємодіє з іншими захисними системами. СВВ можуть на основі результатів своєї роботи оновлювати чорні списки міжмережевих екранів, заносячи туди *IP* адреси машин, запідозрених у здійсненні атаки зловмисників.

На магістральній мережі може бути розміщений досить потужний сервер із встановленим *NIDS*, що дозволяє контролювати весь трафік. Також можна налаштувати відносно невеликі системи для контролю трафіку на певних мережевих вузлах (сервері, комутаторі, шлюзі, маршрутизаторі). [7]

Окрім моніторингу вхідного та вихідного мережевого трафіку, сервер із встановленим *NIDS* може також сканувати системні файли, своєчасно виявляти ненормальну активність та підтримувати дані про цілісність файлів та реагувати на зміни в основних компонентах сервера. Також можна сканувати файли журналів, виявляти підозрілий трафік або незвичну модель поведінки для користувача, вказуючи на можливий компроміс у мережі або спроби віддаленого злому.

NIDS також має здатність грати активну роль, а не тільки оборонну. До програм належать: сканування локальних брандмауерів та мережевих серверів для виявлення потенційних вразливих місць або сканування поточного трафіку. Це може бути корисно для отримання інформації про те, що насправді відбувається в мережі.

Незважаючи на вищесказане, сервер *NIDS* не може замінити основну лінію безпеки, яка складається з брандмауерів, шифрування та антивірусних інструментів. Дуже часто *NIDS* служать допоміжними пристроями для підтримки цілісності мережі. Жодна система *IDS* або система запобігання вторгненню не може повністю замінити загальні заходи безпеки, такі як установки фізичної безпеки, політика корпоративної безпеки тощо. [7]

1.6. Аналіз ринку існуючих систем виявлення вторгнень

На ринку продуктів та послуг з інформаційної безпеки можна знайти велику кількість всіх типів систем, починаючи від невеликих програмних проектів із відкритим кодом і закінчуючи системами захисту інформації від великих виробників. У той же час виробники деяких продуктів захисту інформації роблять наступне: додаткові модулі, які виконують функції *IDS*, встановлюються в продукт захисту інформації, який виконує роль брандмауера. Як правило, вони мають меншу функціональність і продуктивність, оскільки вони не є повноцінною автономною системою. [4]

На даний момент був здійснений порівняльний аналіз СВВ на ринку інформаційної безпеки, який можна знайти у відкритому доступі або придбати у виробника. Результати приведені в таблиці 1.6.1. [4, 17]

Як приклади можна навести програмні пакети, які є безкоштовними та здатними вирішити багато проблем для виявлення вторгнень: *Snort, Suricata, Easy IDS, Bro, Open Source Tripwire*. У контексті цих СВВ *Snor*t та *Suricata* виділяються тим, що насправді вони є стандартами для СВВ. *Suricata* є більш гнучким, ніж *Snort*. Однак багато постачальників розробляють та впроваджують IDS на основі системи *Snort*, не приховуючи цього, а натомість відкрито заявляючи про це. До цих виробників належать *Source Fire, Cisco, FORPOST*. [4]

Таблиця 1.6.1 – Порівняльний аналіз існуючих СВВ

Продукт /Критерій	ПЗ/ПАК	Тип сенсору	Спосіб збору даних	Система аналізу результатів	Сертифікація	Повноцінна документація	Технічна підтримка	Вартість
KFSensor	ПЗ	*HIDS*	Сигнатурний	Немає	Немає	Немає	Немає	120$/міс
OSSEC HIDS	ПЗ	*HIDS*	Сигнатурний	Немає	Немає	Є	Немає	Безкоштовно
Snort	ПЗ	*HIDS*	Сигнатурний	Немає	Немає	Є	Немає	Безкоштовно
Suricata	ПЗ	*HIDS/ NIDS*	Сигнатурний	Немає	Немає	Є	Немає	Безкоштовно
EasyIds	ПЗ	*NIDS*	Сигнатурний	Немає	Немає	Немає	Немає	Безкоштовно
Континент	ПАК	*NIDS/ HIDS*	Сигнатурний/Евристичний	Є	Є	Є	Є	Від 35000 гривень
Cisco IPS	ПАК	*NIDS/ HIDS*	Сигнатурний/Евристичний	Є	Є	Є	Є	Від 65000 гривень
Рубикон	ПЗ/ПАК	*NIDS/ HIDS*	Сигнатурний/Евристичний	Є	Є	Є	Є	Від 34000 гривень

Продовження таблиці 1.6.1

Ручей-М	ПАК	*NIDS/ HIDS*	Сигнатурний/Евристичний	Є	Є	Є	Є	Платно
ViPNet IDS	ПАК	*NIDS/ HIDS*	Сигнатурний/Евристичний	Є	Є	Є	Є	Від 60000 гривень
Форпост	ПЗ/ПАК	*NIDS/ HIDS*	Сигнатурний/Евристичний	Є	Є	Є	Є	Платно
Bro	ПЗ	*NIDS*	Сигнатурний	Немає	Немає	Є	Немає	Безкоштовно
Prelude Hybrid IDS	ПЗ	*NIDS/ HIDS*	Сигнатурний/Евристичний	Немає	Немає	Є	Є	Платно
Samhain	ПЗ	*HIDS*	Сигнатурний	Немає	Немає	Є	Є	Безкоштовно
Open Source Tripwir	ПЗ	*NIDS/ HIDS*	Сигнатурний	Немає	Немає	Є	Є	Безкоштовно
Check Point IPS	ПЗ	*NIDS/ HIDS*	Сигнатурний/Евристичний	Є	Немає	Є	Є	Платно
McAfee IPS	ПЗ/ПАК	*NIDS/ HIDS*	Сигнатурний/Евристичний	Є	Є	Є	Є	Платно

Продовження таблиці 1.6.1

IBM ISS Proventia IPS	ПЗ/ПАК	*NIDS/ HIDS/ APIDS*	Сигнатурний/Евристичний	Є	Є	Є	Є	Платно
OSSIM	ПЗ	*HIDS*	Сигнатурний/Евристичний	Є	Є	Є	Є	1075$/міс
Sourcefire IPS	ПЗ/ПАК	*NIDS/ HIDS*	Сигнатурний/Евристичний	Є	Є	Є	Є	Платно

РОЗДІЛ 2. РОЗРОБКА ФУНКЦІОНАЛЬНОЇ МОДЕЛІ ВІРТУАЛЬНОГО СТЕНДУ

2.1. Вибір платформи віртуалізації

В даний час існує обмежений вибір платформ для віртуалізації. Основними постачальниками цих послуг є *VMware* та *Oracle*. Решта постачальників пропонують менше функціональних можливостей або припинили випуск нових продуктів.

VMware Player (Work station) - ця платформа в повному функціоналі є на платному ринку, але в той же час існує неповна безкоштовна версія. Це чудовий інструмент для розробників програмного забезпечення та адміністраторів мережі. Вона має технологію *VMware MultipleWorld*, яка забезпечує ізоляцію операційних систем у створених віртуальних машинах, полегшує оновлення або встановлення операційних машин без необхідності перезавантажувати комп'ютер або маніпулювати розділами диска. [12]

Oracle Virtual Box - це безкоштовна платформа з відкритим кодом. Підтримує багато операційних систем, придатна для тестування, розробки та впровадження декількох операційних систем на одному комп'ютері. [13]

Порівняльна характеристика віртуальних машин буде приведена в таблиці 2.1.

Таблиця 2.1 – Порівняння характеристик віртуальних машин

	Oracle VirtualBox	*VMware Player*
Хостова ОС	*Linux, Solaris, Windows, OpenSolaris, Mac OS X, FreeBSD*	*Windows, Linux*
ОС які підтримуються	*DOS, Windows, Solaris, Linux, OpenSolaris, FreeBSD, OpenBSD, Net-BSD, Netware, Mac OS X*	*DOS, Windows, Solaris, Linux, OpenSolaris, FreeBSD, Netware*
Кількість снапшотів	Необмежена	32
Гостьовий процесор	Такий як і на хостовій ОС	*Intel x86, AMD64*
Типове використання	Технічні спеціалісти, тестувальники, розробники	
Швидкість роботи	Швидкість роботи практично без втрат в порівнянні з хостовою ОС, якщо використовуються розширення	
Ліцензія	Вільна і пропріетарна	Пропрієтарна

Основними вимогами до платформи віртуалізації для вирішення наших завдань є: вартість продукту, перелік підтримуваних операційних систем (ОС) та зразки віртуальних машин. В цілому обидві платформи чудово підходять для розробки віртуально стенду, проте було прийняте рішення використовувати *Oracle VM VirtualBox* через те що він безкоштовний.

2.2. Визначення критеріїв для вибору

Системи виявлення вторгнень впроваджені в навчальних цілях, тому першою вимогою є безкоштовне використання в освітніх цілях.

Оскільки робота з системою в майбутньому проводитиметься студентами, потрібен такий продукт, використання якого не передбачає проходження

платних навчальних курсів для роботи з системою. Потрібно лише використовувати існуючу документацію. З тієї ж причини система повинна мати можливість зберегти власну конфігурацію або стан системи, або мати систему профілів для збереження конфігурації.

Система повинна мати не тільки систему аналізу трафіку, але й систему аналізу результатів роботи. Для практичного використання вона також повинна мати графічний інтерфейс.

Розгортання та встановлення системи не повинно обтяжуватися додатковими фінанСВВими витратами, тоді як система повинна мати можливість масштабуватися з потенційним збільшенням обчислювальної потужності мережі.

Продуктивність системи не повинна відставати від продуктивності локальної мережі, на якій відбувається розгортання.

Впроваджена СВВ не повинна вносити будь-яких змін у роботу існуючої локальної мережі, виконуючи при цьому повноцінно свої функції моніторингу трафіку та виявлення вторгнень.

На основі цих вимог та проведеного аналізу були визначені такі критерії вибору системи виявлення вторгнень:

- програмне забезпечення;
- мережева система виявлення вторгнень;
- сигнатурний метод підпису аналізу даних;
- безкоштовне розповсюдження;
- *open source*;
- наявність документації;
- система аналізу результатів;

2.3. Аналіз системи виявлення вторгнень *Snort*

Snort - це безкоштовне програмне забезпечення з відкритим кодом під ліцензією *GPL*. Спочатку *Snort* був створений одним з найвідоміших людей у

світі інформаційної безпеки, автором багатьох книг, Мартіном Роше в 1998 році [18]. Головною причиною створення цієї СВВ була відсутність інструменту сповіщення про досить ефективні атаки, особливо безкоштовної, на той час. [8]

Snort може виявляти наступне:

- поганий трафік;
- використання експлойтів;
- сканування системи (порти, ОС);
- атаки на такі служби як *FTP*, *Telnet*, *DN*S, і тд.;
- атаки *DoS/DDoS*;
- атаки, пов’язані з *WEB* серверами;
- атаки на бази даних *SQL, Oracle* та ін.;
- атаки о протоколам *SNMP, Net bios*, *ICMP*;
- атаки на *SMTP, imap, pop2, pop3*;
- різні *Back doors;*
- *web* – фільтри (порнографія).

Snort працює наступним чином (рисунок 2.1)

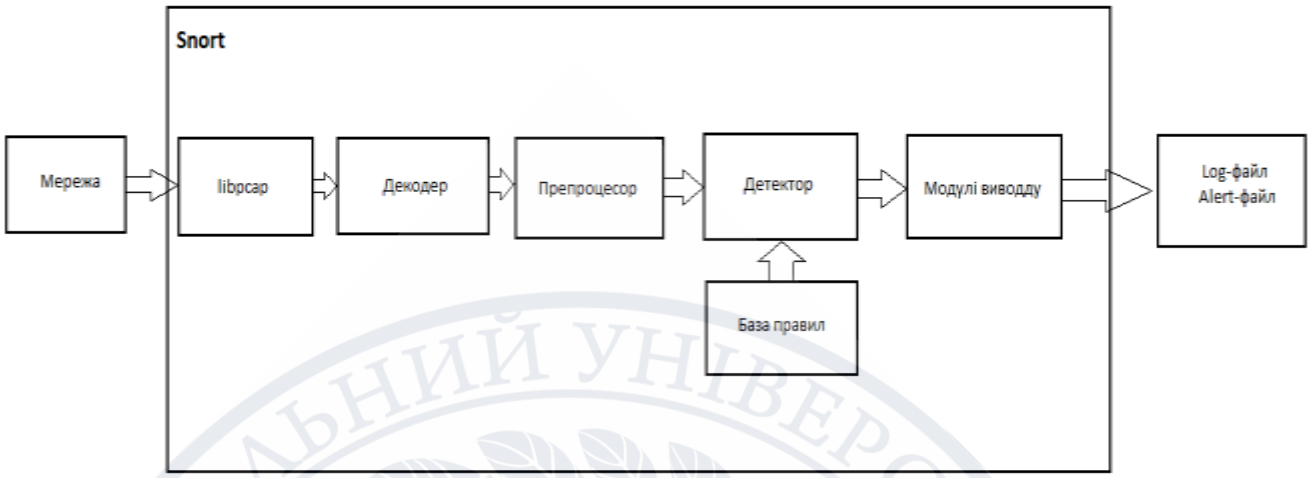

Рисунок 2.1 – принцип роботи *Snort*

Схема роботи включає в себе:

- бібліотеку *libpcap*;
- декодер пакетів;
- препроцесор;
- детектор;
- база правил;
- модулі виводу інформації.

Бібліотека *libpcap* дозволяє захоплювати пакети, що надходять на мережеву карту, перед тим, як вони потрапляють у стек протоколів. На основі цієї бібліотеки створюються програми для моніторингу та тестування мережі, такі як *Snort*, а також аналізатори трафіку, наприклад, *Wire Shark*. [8]

Після перехоплення пакет потрапляє в декодер, його завданням є деінкапсуляція даних із мережі та транспортних рівнів (*IP, TCP, UDP, ICMP*) протоколів рівня каналу передачі даних, таких як *Ethernet* або 802.11.

Препроцесор готує дані, отримані від протоколів транспортного та мережевого рівня, для подальшої обробки детектором. У *Snort* існує конфігурація препроцесорів та їх правила, що загалом дозволяє збільшити швидкість роботи системи.

Детектор аналізує дані, отримані шляхом пошуку в пакетах певних правил та підписів у базі даних. Самі правила складаються з опису правила, підпису, опису загрози та реакції при виявленні.

Проаналізувавши дані, *Snort* відображає необхідну інформацію (*Log, Alert*) у необхідних форматах.

Snort може працювати у трьох режимах:

- режим аналізатора трафіку;
- режим реєстрації пакетів;
- режим виявлення вторгнень.

Barnyard2

Barnyard2 – це інтерпретатор двійкових файлів з відкритим кодом для файлів, створених *Snort*. [8]

Стандартний спосіб запису подій, наданий *Snort* на консоль або у файл, вимагає великих ресурсів. В ідеалі події *Snort* слід зберігати в базі даних *MySQL*, щоб можна було знаходити та переглядати потрібні події. Barnyard2 буде використовуватися для відображення подій *Snort* у базі даних *MySQL*. *Snort* налаштовано на передачу подій у двійковому вигляді до папки, а потім Barnyard2 зчитує ці події та вставляє їх у базу даних *MySQL*.

PulledPork

PulledPork - це сценарій, який завантажує, комбінує, встановлює та оновлює правила для *Snort* з різних джерел. Існує кілька наборів правил, які *PulledPork* може завантажити. Його можна налаштувати для завантаження набору правил спільноти *Snort* безкоштовно, без створення безкоштовного облікового запису на *Snort.org*. [8]

2.4. Аналіз системи виявлення вторгнень Suricata

Як і *Snort*, *Suricata* складається з декількох модулів (захоплення, збір, декодування, виявлення та виведення), за замовчуванням перед декодуванням

захоплений трафік йде одним потоком, це оптимально з точки зору виявлення, але він більше завантажує систему. На відміну від *Snort*, у *Suricata* може перевизначити поведінку трафіку за допомогою налаштувань і буквально за допомогою одного налаштування розділити потоки відразу після захоплення, а за допомогою іншого можна вказати, як потоки розподілятимуться між процесорами. Це забезпечує широкі можливості для оптимізації обробки трафіку на конкретних комп'ютерах у певній мережі. [9, 19]

Suricata підтримує вилучення та перевірку файлів, переданих через HTTP, аналіз стисненого вмісту, можливість ідентифікації за *URI*, файлами *cookie*, заголовками, *user-agent*, тілом запиту та відповіддю. *Suricata* використовує цю функцію в деяких мережах для реєстрації *HTTP*-трафіку без виявлення. Зміст послідовності можна розрізнити за маскою та за допомогою регулярних виразів; файли можна ідентифікувати за іменем, типом або контрольною сумою *MD5*.

Розшифрування IPv6 підтримується власним способом, включаючи *IPv4-in-IPv6, IPv6-in-IPv6*, тунелі *Teredo* та деякі інші. Модульна конструкція движка дозволяє швидко підключити новий елемент для захоплення, декодування, аналізу або обробки пакетів. Для перехоплення трафіку використовується кілька інтерфейсів: *NF Queue, IPF Ring, Lib Pcap, IPFW, AF_PACKET, PF_RING*. Режим *Unix Socket* дозволяє автоматично сканувати файли *PCAP*, раніше захоплені іншою програмою (наприклад сніфером).

У *Snort* режим *IPS* з'явився не відразу, але в *Suricata* режим блокування зловмисного трафіку реалізований нестандартно і виконується за допомогою стандартного фільтра пакетів операційної системи. Наприклад, *Linux* використовує два режими IPS: через чергу *NFQUEUE*, яку можна обробити на рівні користувача, та через режим *AF*_PACKET з нульовою копією (представлений у версії 1.4). Режим AF_*PACKET* дуже швидкий, але вимагає двох мережевих інтерфейсів, система повинна працювати як шлюз. Якщо пакет заблоковано, він просто не перенаправляється на другий інтерфейс. У випадку *NFQUEUE* алгоритм простіший. Після того, як пакет потрапляє в

iptables, він надсилається до черги *NFQUEUE*, де виконується згідно з правилами. Можуть вийти три дії: *NF_ACCEPT, NF_DROP та NF_REPEAT*. Останнє дозволяє маркувати пакети, а потім виконувати їх згідно з наступними таблицями / правилами *iptables*. [9]

Головною особливістю *Suricata* є те, що, крім унікальних розробок, він використовує майже все, що вже розроблено для *Snort*. Тому підходять всі набори рулсетів *Snort - Sourcefire VRT, OpenSource Emerging Threats (ETOpen)* та *Commercial Emerging Threats Pro*. Вихід є уніфікованим (*Unified2*), тому результат можна проаналізувати за допомогою звичайних бекендів: *Barnyard2, Snortsnarf, Snorby, Aanval, BASE, FPCGUI, NSM, Sguil і Squert*. Можливий вихід на *PCAP, Syslog*, файли тощо. Наприклад, *Suricata* веде реєстр ключів та сертифікатів, які відображаються у з'єднаннях *TLS / SSL*. В останніх версіях з'явився журнал *Eve*, який виводить події у форматі *JSON* для попереджень. Наявність *JSON* значно спрощує інтеграцію *Suricata* зі сторонніми програмами, включаючи системи відображення журналів та моніторингу (наприклад, *Kibana*). [9, 10]

Всі налаштування та правила *Suricata* виконуються у файлах *YAML*, це більш наочно та спрощує автоматичну обробку. Однією з переваг *Suricata* є обробка 7-го рівня *OSI*, що покращує її здатність виявляти зловмисне програмне забезпечення. Механізм автоматично виявляє та аналізує протоколи (*IP, TCP, UDP, ICMP, HTTP, TLS, FTP, SMB, SMTP* та інші), тому в правилах він не може бути строго прив'язаний до номера порту, як це робиться в *Snort* , просто вказується протокол та дія. Крім того, самі модулі *Suricata* вже обробляють трафік і виявляють протокол, навіть якщо використовується нестандартний порт.

Правила містять компоненти: дія (передача, скидання, відхилення або попередження), заголовок (джерело та адреса *IP* / порт призначення) та опис (що шукати). У базовій конфігурації є небагато правил, а деякі також відключені (коментуються) у самих файлах, тому наразі слід більше зосередитися на тих, які були підключені до *Snort* за допомогою *PulledPork*.

Іноді між вузлами відкривається не одне, а кілька з'єднань *TCP*. Деякі *IDS* не бачать загальної картини і обробляють кожен потік окремо. Правила *Suricata* широко використовують поняття потокових бітів. Для відстеження кількості тригерів правила використовуються різні змінні сеансу (наприклад, за допомогою *flowint*), щоб створити лічильники та прапори, а потім перевірити їх. Цей підхід легко справляється зі спробою грубого введення пароля. У версії 2.1 можна легко простежити правила *IP*-джерела - призначення *IP*-пакетів (*xbits*), що робить ще простішим виявлення зловмисного трафіку, що поширюється по кількох з'єднаннях. [9, 10]

В останніх версіях з’явилася підсистема репутації *IP*, яка завантажує та використовує дані з різних баз даних, що містять у правилах списки репутації хоста. Спеціальний механізм забезпечує швидкий пошук та відповідність *IP*-адресам.

ELK

Elasticsearch - це пошукова машина *json rest api*, яка використовує *Lucene* і написана на *Java*. Цю програму можна використовувати для складних пошуків у базах даних документів, у випадку *Suricata Elasticsearch* знайде та обробить журнали подій. [10]

Logstash - це програма для збору, фільтрації та перенаправлення до цільового сховища даних. [10]

Kibana - це сервіс для віртуалізації та перегляду даних *Elasticsearch*. Ця програма створює інформаційні панелі, налаштовує форму відображення, формує інтерактивні діаграми, а також надає геодані. [10]

2.5. Створення віртуального стенду

Для створення віртуального стенду в програмі *VirtualBox* потрібно створити клієнтський комп’ютер і комп’ютер який комп'ютер зловмисника.

Для цього знадобиться встановити операційну систему *Ubuntu* для клієнтського комп’ютера, і *Kali Linux* для комп’ютера зловмисника.

Віртуальні машини показані на рисунку 2.6.1.

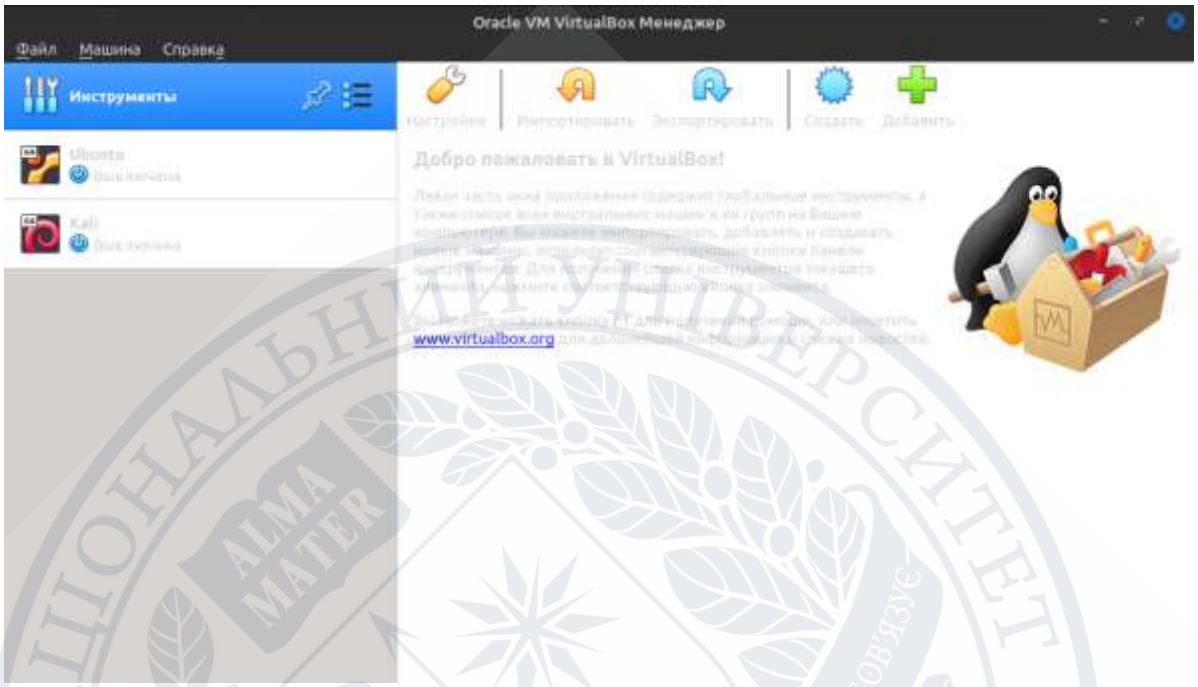

Рисунок 2.6.1 – Віртуальні машини

Назви віртуальних машин, логін, пароль і IP-адреси показані в таблиці 2.6.1.

Таблиця 2.6.1 – Таблиця користувачів

Назва	Логін	Пароль	IP-адреса
Ubuntu	*mamuk*	1	192.168.0.107
Kali	*mamuk*	1	192.168.0.109

VirtualBox дозволяє налаштовувати мережеві адаптери для спільної роботи.

На клієнтському комп’ютері встановлена *Ubuntu* 20.04, на комп’ютері зловмисника *Kali Linux* 2021.1.

Безкоштовно скачати дистрибутиви можна з офіційного сайтів *https://www.kali.org/downloads/* та *https://ubuntu.com/*.

Мережеві інтерфейси клієнтського комп’ютера та комп’ютера зловмисника налаштовані в режимі «Сетевой мост».

РОЗДІЛ 3. РОЗРОБКА ВІРТУАЛЬНОГО СТЕНДУ

3.1. Налаштування віртуального стенду

Перша віртуальна машина з робочою назвою «*Ubuntu*». Дана машина буде виконувати роль клієнтського комп'ютера який не обхідно захищати. На неї встановлюється ОС *Ubuntu* 20.04.

Друга машина з робочою назвою *Kali*. Дана машина буде призначена для дій зловмисника. На неї встановлюється ОС *Kali Linux* 2021.1.

Для взаємодії віртуальних машин потрібно мережеві адаптери обох машин налаштувати в режимі «Сетевой мост», для того щоб їм були присвоєні IP-адреси і вони мали доступ в інтернет.

3.2. Встановлення та налаштування системи виявлення вторгнень *Snort*

Перш ніж починати налаштовувати *Snort*, потрібно підготувати віртуальну машину, а також вибрати мережевий інтерфейс. Мережевий інтерфейс *enp0s3*, який належить до мережі 192.168.0.107, був обраний для роботи. Надалі в цій мережі можна буде створювати більше клієнтських комп'ютерів, не змінюючи налаштувань *Snort*. Крім того, перед тим, як розпочати встановлення *Snort*, потрібно встановити необхідні реквізити.

Snort має деякі основні реквізити та деякі другорядні. В рамках роботи вони будуть встановлювати всі реквізити. Перш за все, потрібно встановити всі інструменти для створення програмного забезпечення. Усі реквізити встановлюються із репозиторіїв *Ubuntu*: [14]

$ sudo apt-get install -y build-essential libpcap-dev libpcre3-dev libdumbnet-dev bison flex zlib1g-dev liblzma-dev openssl libssl-dev libnghttp2-dev.

Для зручності, створюється папка в якій будуть зберігатисяфайли встановлення.

$ *mkdir snort_tmp*

$ *cd snort_tmp*

$ *wget https://snort.org/downloads/snort/daq-2.0.6.tar.gz*

$ *tar -xvzf daq-2.0.6.tar.gz*

$ *cd daq-2.0.6*

$ *./configure*

$ *make*

$ *sudo make install*

Після встановлення всіх необхідних реквізитів можна приступити до встановлення *Snort*. Для цього потрібно завантажити оригінальний файл *Snort*, скомпілювати його та встановити. Опція *"--enable-sourcefire*" забезпечує моніторинг продуктивності пакетів для моніторингу правил, виконання препроцесора та роботи з *Snort* так як команда розробників:

$ *cd ~/snort_src*

$ *wget https://www.snort.org/downloads/snort/snort-2.9.12.tar.gz*

$ *tar -xvzf snort-2.9.12.tar.gz*

$ *cd snort-2.9.12*

$ *./configure --enable-sourcefire --disable-open-appid*

$ *make*

$ *sudo make install*

$ *sudo ldconfig*

$ *sudo ln -s /usr/local/bin/snort /usr/sbin/snort*

Потрібно запустити команду *Snort* із прапором *–V (snort -V).* Це дозволить *Snort* перевірити себе, свою версію та файли конфігурації. Під час виконання команди на екрані відображатиметься результат, схожий на той, що показаний на рисунку 3.3.1.

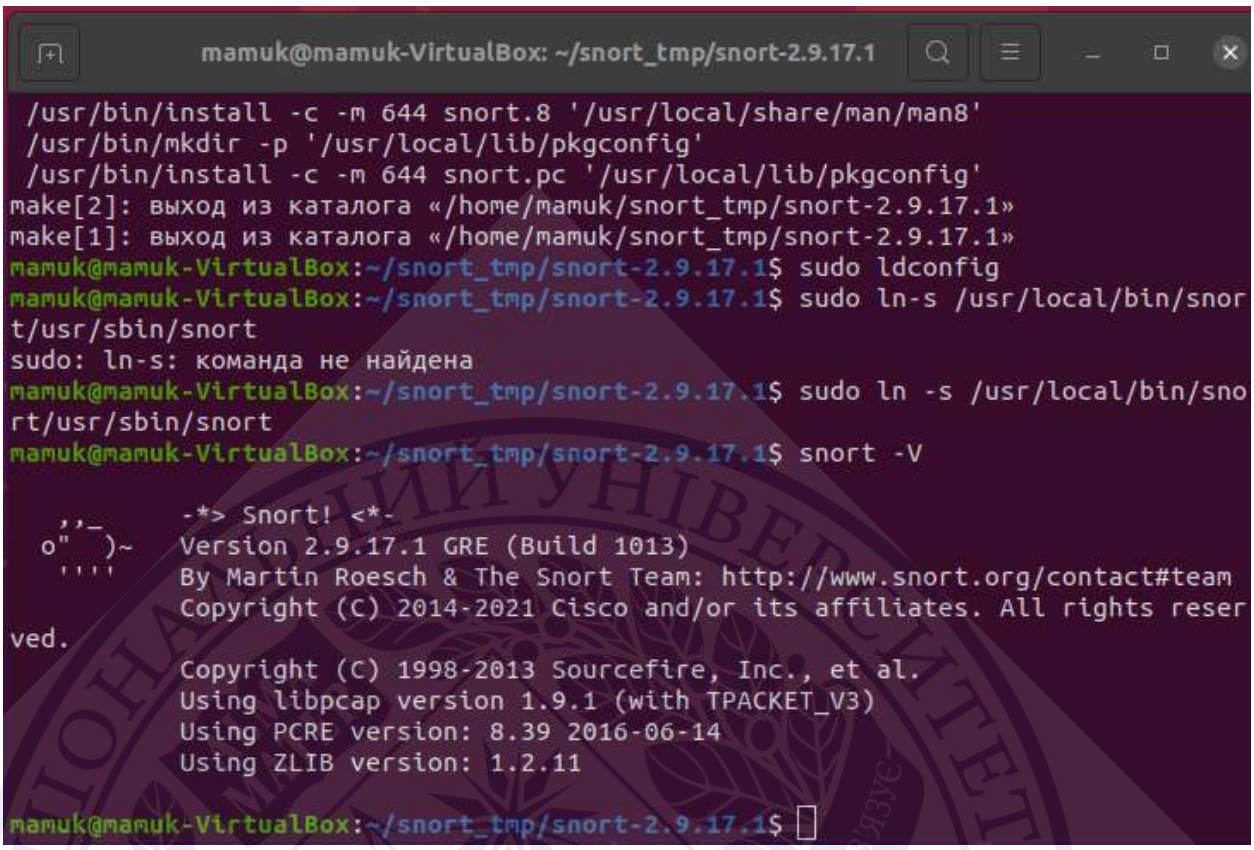

Рисунок 3.3.1 – Результат виконання команди

Після успішного встановлення *Snort* потрібно створити непривілейований обліковий запис та групу (*snort:snort*). Також потрібно створити деякі файли та каталоги, які вимагає *Snort*, і встановити для них дозволи. Потрібно створити кілька каталогів, для роботи *Snort.*

Створення групи та облікового запису:

$ *sudo groupadd snort*

$ *sudo useradd snort -r -s /sbin/nologin -c SNORT_IDS -g snort*

Створення директорій для *Snort*:

$ *sudo mkdir /etc/snort*

$ *sudo mkdir /etc/snort/rules*

$ *sudo mkdir /etc/snort/rules/iplists*

$ *sudo mkdir /etc/snort/preproc_rules*

$ *sudo mkdir /usr/local/lib/snort_dynamicrules*

$ *sudo mkdir /etc/snort/so_rules*

Створення файлів для збереження даних та *IP*-адресів:

$ *sudo touch /etc/snort/rules/iplists/black_list.rules*

$ *sudo touch /etc/snort/rules/iplists/white_list.rules*

$ *sudo touch /etc/snort/rules/local.rules*

$ *sudo touch /etc/snort/sid-msg.map*

Створення директорій для зберігання лог-файлів:

$ *sudo mkdir /var/log/snort*

$ *sudo mkdir /var/log/snort/archived_logs*

Налаштування доступу:

$ *sudo chmod -R 5775 /etc/snort*

$ *sudo chmod -R 5775 /var/log/snort*

$ *sudo chmod -R 5775 /var/log/snort/archived_logs*

$ *sudo chmod -R 5775 /etc/snort/so_rules*

$ *sudo chmod -R 5775 /usr/local/lib/snort_dynamicrules*

Налаштування прав доступу для *Snort*:

$ *sudo chown -R snort:snort /etc/snort*

$ *sudo chown -R snort:snort /var/log/snort*

$ *sudo chown -R snort:snort /usr/local/lib/snort_dynamicrules*

Далі необхідно скопіювати файли конфігурації в створені директорії:

$ *cd ~/snort/snort-2.9.12/etc/*

$ *sudo cp *.conf* /etc/snort*

$ *sudo cp *.map /etc/snort*

$ *sudo cp *.dtd /etc/snort*

$ *cd~/snort/snort-2.9.12/src/dynamic-preprocessors/build/usr/local/lib/snort_dynamicpreprocessor/*

$ *sudo cp * /usr/local/lib/snort_dynamicpreprocessor/*

В результаті повинна вийти така структура файлів та папок яка зображена на рисунку 3.3.2.

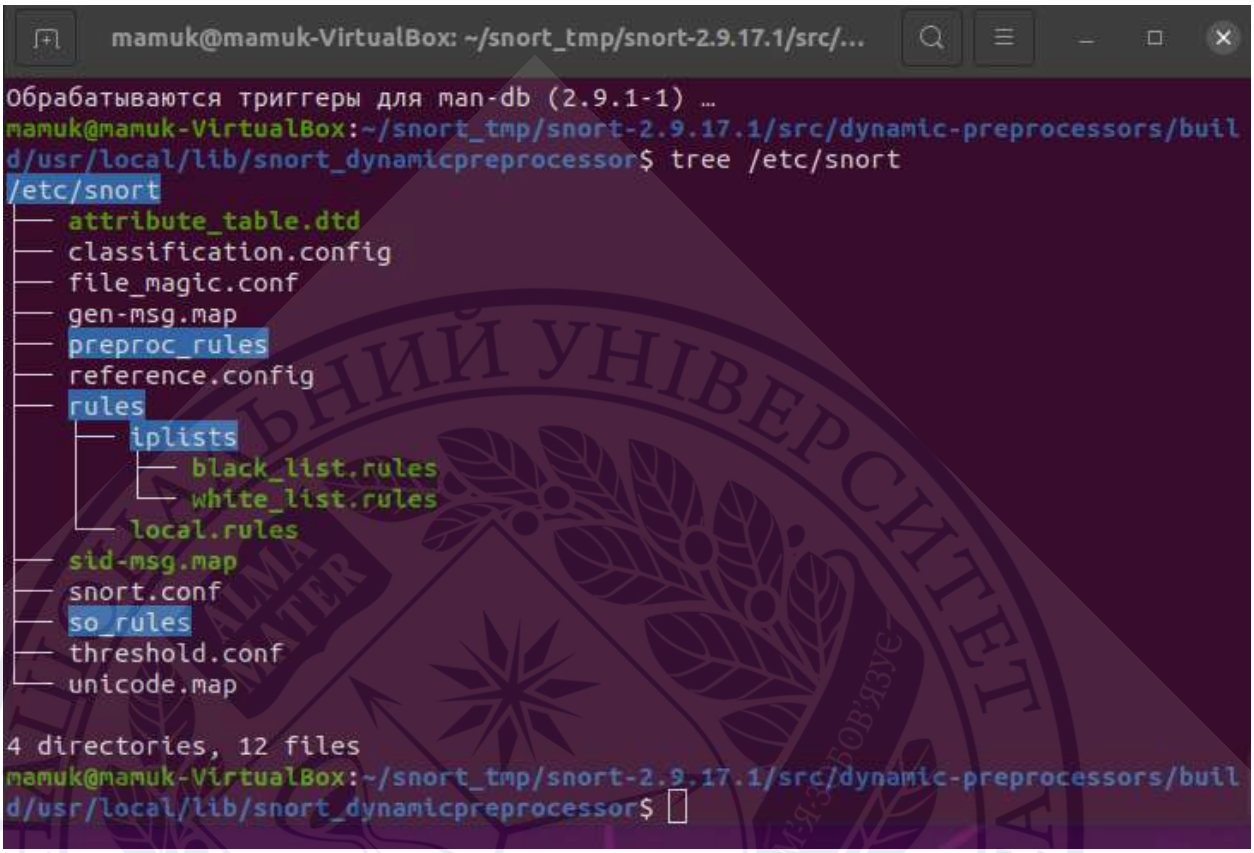

Рисунок 3.3.2 – Структура файлів та папок

Наступним кроком є редагування основного конфігураційного файлу *Snort* (*/etc/snort/snort.conf*). Коли *Snort* запускається, цей конфігураційний файл повідомляє програмі запускатися у певному режимі. Цей файл керуватиме наборами правил *Snort*. [14]

Оскільки *PulledPork* буде використовуватися для завантаження правил, потрібно закоментувати деякі рядки у файлі конфігурації. Ці рядки відповідають за підключення правил:

$ *sudo sed -i 's/include \$RULE_PATH/#include \$RULE_PATH/' /etc/snort/snort.conf*

Далі потрібно вручну змінити деякі радки в файлі */etc/snort/snort.conf:*

$ *sudo gedit /etc/snort/snort.conf*

- Стрічка 45: вказуємо нашу мережу.
- Стрічка 104: змінюємо на var *RULE_PATH /etc/snort/rules.*

- Стрічка 105: змінюємо на *var SO_RULE_PATH /etc/snort/so_rules*.
- Стрічка 106: змінюємо на *var PREPROC_RULE_PATH /etc/snort/preproc_rules.*
- Стрічка 113: змінюємо на *var WHITE_LIST_PATH /etc/snort/rules/iplists*.
- Стрічка 114: змінюємо на *var BLACK_LIST_PATH /etc/snort/rules/iplists.*
- Стрічка 546: необхідно розкоментувати include *$RULE_PATH/local.rules* для можливості використовувати власні правила.
- Стрічка 521: нижче потрібно добавити строку *output unified2: filename snort.u2, limit 128.*
- Стрічка 548: необхідно добавити строку *include $RULE_PATH/snort.rules.*

Після завершення редагування файлу конфігурації слід перевірити правильність усіх змін. Для перевірки працездатності буде використано прапор *–T*, для перевірки файлу конфігурації прапор –c, для вказівки *Snort*, який файл конфігурації використовувати прапор та прапор –i для вказівки на якому інтерфейсі буде прослуховуватися *Snort.*

$ *sudo snort -T -c /etc/snort/snort.conf -i enp0s3*

Результат виконання команди приставлений на рисунку 3.3.3.

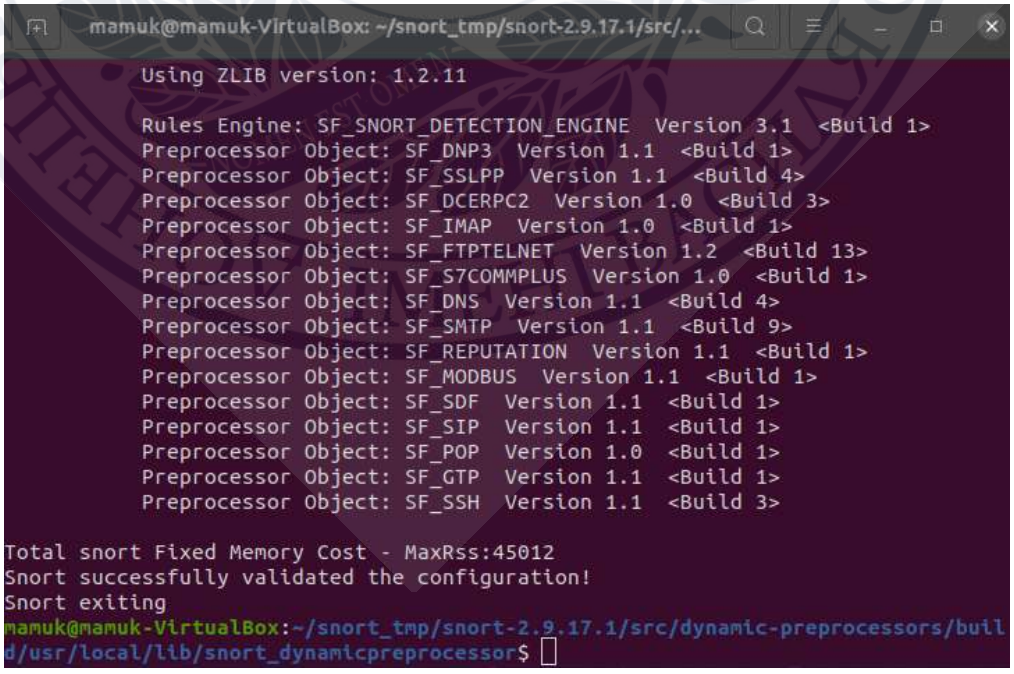

Рисунок 3.3.3 – Результат перевірки Snort

Отриманий результат показує що всі зміни в файлі були зроблені правильно, і *Snort* готовий для подальшої роботи.

Далі потрібно перевірити працездатність *Snort*. Для цього потрібно написати просте правило яке буде відслідковувати *ICMP* пакети. Посилати *ICMP* пакети будемо з віртуальної машини зловмисника з допомогою команди *ping*. [14]

Потрібно відкрити правило:

$ *sudo gedit /etc/snort/rules/local.rules*

І написати правило:

alert icmp any any -> $HOME_NET any (msg:"ICMP packet detected"; GID:1; sid:10000001; rev:001; classtype:icmp-event;)

Barnyard2 не читає метадані, попередження з файлу *local.rules*. Без цієї інформації він не будете знати жодної інформації про правило, яке спричинило попередження, і генеруватиме не фатальні помилки при додаванні нових правил за допомогою *PulledPork*. Щоб переконатися, що *barnyard2* знає, що правило, яке було створено з унікальним ідентифікатором 10000001 і має повідомлення " *ICMP test detected*", потрібно добавити в файл */etc/snort/sid-msg.map* строку:

1 || 10000001 || 001 || icmp-event || 0 || ICMP Test detected || url,tools.ietf.org/html/rfc792

Оскільки раніше вносилися зміни у файл конфігурації, потрібно переконатися, що зміни правильні. Також можна побачити, що створене правило було успішно завантажено, рисунок 3.3.4.

Перевірка конфігурації:

$ *sudo snort -T -c /etc/snort/snort.conf -i enp0s3*

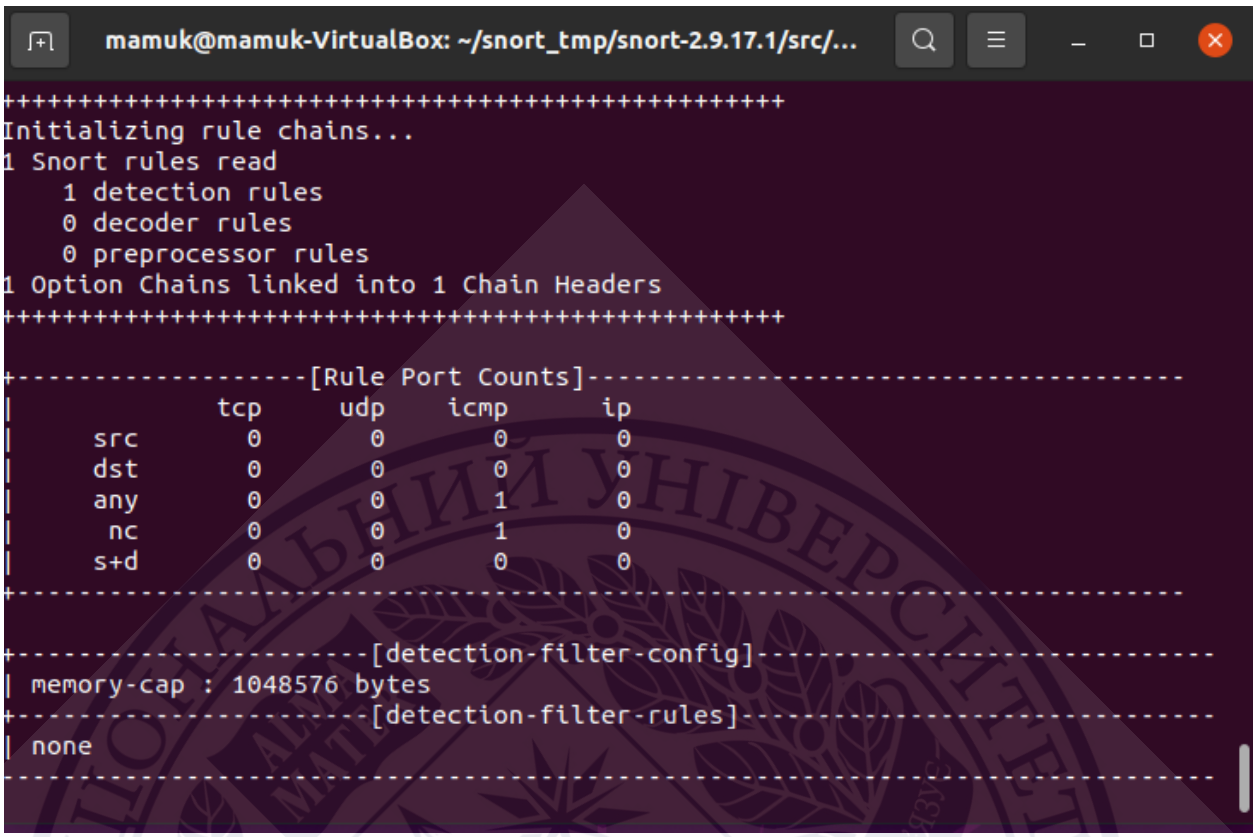

Рисунок 3.3.4 – Нове правило завантажено

Тепер можна запустити *Snort*, використовуючи наступні прапори:

– *-A* – виведення даних в термінал;

– *-q* – виведення вікна стану;

– *-u snort* – запуск під користувачем;

– *-g snort* – запуск під групою.

$ *sudo /usr/local/bin/snort -A console -q -u snort -g snort –c /etc/snort/snort.conf -i enp0s3*

Після виконання цієї команди інформація в терміналі не відображатиметься, поки пакет не досягне вказаного інтерфейсу, який, порівняно з правилом, буде відповідати йому.

На віртуальній машині «*Kali*» виконується команда *ping* на *IP*-адресу 192.168.0.107. Результат приставлений на рисунку 3.3.5.

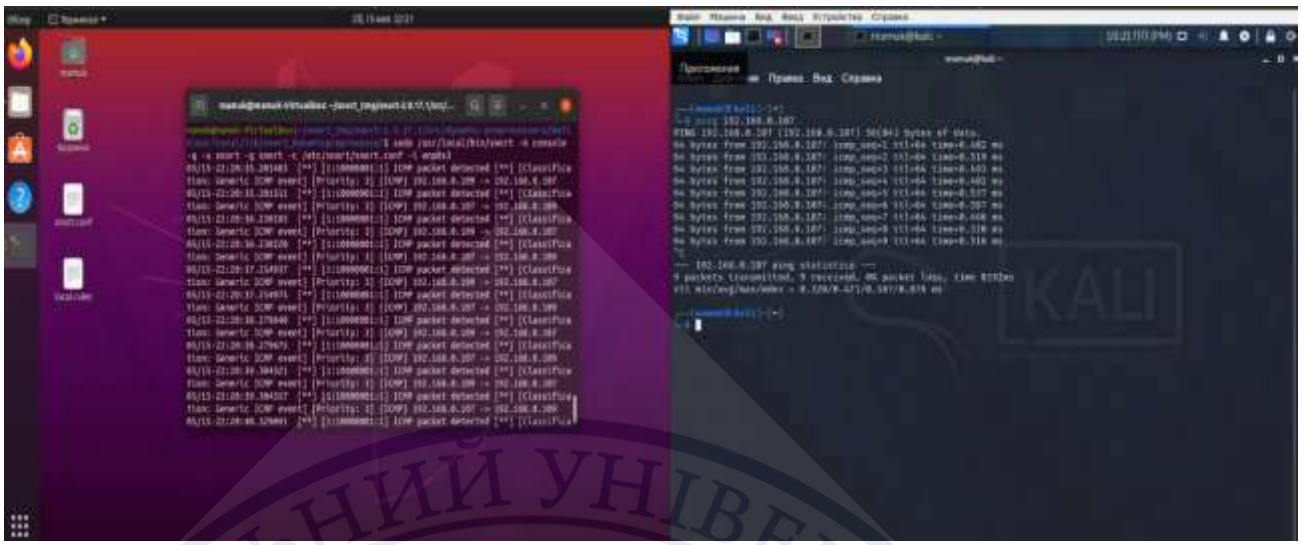

Рисунок 3.3.5 – Результат роботи правила

Можна зупинити *Snort*, скориставшись комбінацією клавіш *Ctrl + C. Snort* зберіг копію цієї інформації у */var/log/snort* під ім'ям *snort.log.nnn*. Це підтверджує, що *Snort* працює належним чином та генерує попередження.

Встановлення Barnyard2.

Вищевказаний спосіб запису подій або на консоль, або у файл вимагає великих ресурсів. В ідеалі події *Snort* слід зберігати в базі даних *MySQL*, щоб можна було знаходити та переглядати потрібні події. *Barnyard2* буде використовуватися для відображення подій *Snort* у базі даних *MySQL*. *Snort* налаштовано на передачу подій у двійковому вигляді до папки, а потім Barnyard2 зчитує ці події та вставляє їх у базу даних *MySQL*. [14]

Перед встановлення *barnyard2* потрібно встановити необхідні реквізити:

$ *sudo apt install -y mysql-server libmysqlclient-dev mysql-client autoconf libtool*

Встановлення *Barnyard2*:

$ *cd snort_src*

$ *wget https://github.com/firnsy/barnyard2/archive/master.tar.gz -O barnyard2-Master.tar.gz*

$ *tar zxvf barnyard2-Master.tar.gz*

$ *cd barnyard2-master*

$ *autoreconf -fvi -I ./m4*

$ *sudo ln -s /usr/include/dumbnet.h /usr/include/dnet.h*

$ *sudo ldconfig*

$ *./configure --with-mysql --with-mysql-libraries=/usr/lib/x86_64-linux-gnu*

$ *make*

$ *sudo make install*

Щоб перевірити чи правильно встановився *barnyard2*, вводиться наступна команда:

$ *sudo /usr/local/bin/barnyard2 –V*

Рисунок 3.3.5 – результат встановлення Barnyard2

Наступним кроком буде копіювання та створення файлів які необхідні barnyard2 для коректної роботи:

$ *sudo cp ~/snort/barnyard2-master/etc/barnyard2.conf /etc/snort/*

$ *sudo mkdir /var/log/barnyard2*

$ *sudo chown snort.snort /var/log/barnyard2*

$ *sudo touch /var/log/snort/barnyard2.waldo*

$ *sudo chown snort.snort /var/log/snort/barnyard2.waldo*

Оскільки Barnyard2 зберігає попередження в базі даних *MySQL*, потрібно створити цю базу даних, а також користувача *MySQL* "*snort*" для доступу до бази даних. Для створення бази даних та користувача *MySQL* потрібно виконати такі команди: [14]

$ *mysql -u root -p*

$ *mysql> create database snort;*

$ *mysql> use snort;*

$ *mysql> source ~/snort/barnyard2-master/schemas/create_mysql*

$ *mysql> CREATE USER 'snort'@'localhost' IDENTIFIED BY 'toor';*

$ *mysql> grant create, insert, select, delete, update on snort.* to 'snort'@'localhost';*

$ *mysql> exit*

Потрібно вказати *Barnyard2*, як підключатися до бази. Потрібно змінити файл */etc/snort/barnyard2.conf* і добавити наступну строку в кінець:

output database: log, mysql, user=snort password=toor dbname=snort host=localhost sensor name=sensor01

Так як пароль зберігається в відкритому вигляді, потрібно обмежити читання даного файлу для інших користувачів.

$ *sudo chmod o-r /etc/snort/barnyard2.conf*

Тепер варто перевірити, чи *Snort* записує події у правильний двійковий файл журналу, і що *Barnyard2* читає ці журнали та пише події в базу даних *MySQL*. Запускаємо *Snort* в режимі попередження:

$ *sudo /usr/local/bin/snort -q -u snort -g snort -c /etc/snort/snort.conf –I enp0s3*

Використовуючи віртуальну машину "*Kali*", знову проводимо пінг *IP*-адреси 192.168.0.107. У цьому випадку на екрані не буде виводу, оскільки *Snort* був запущений без позначки *-A*. Відправляючи кілька пакетів, потрібно перервати *ping* і зупинити *Snort*, натиснувши *Ctrl + C*. У цьому випадку все збережеться у файлі, створеному в каталозі */var/log/snort* з ім'ям "*snort.u2.nnn*".

Тепер вам потрібно запустити *Barnyard2* для обробки подій у "*snort.u2.nnn*" та завантаження їх у базу даних *Snort*. Для цього використовується команда:

$ *sudo barnyard2 -c /etc/snort/barnyard2.conf -d /var/log/snort -f snort.u2 -w /var/log/snort/barnyard2.waldo -g snort -u snort*

Запускається *Barnyard2* та починає обробляти попередження в файлі */var/log/snort/snort.u2.nnn,* починається виведення результатів на екран та запис в базу даних. Результат роботи приставлений на рисунку 3.3.6.

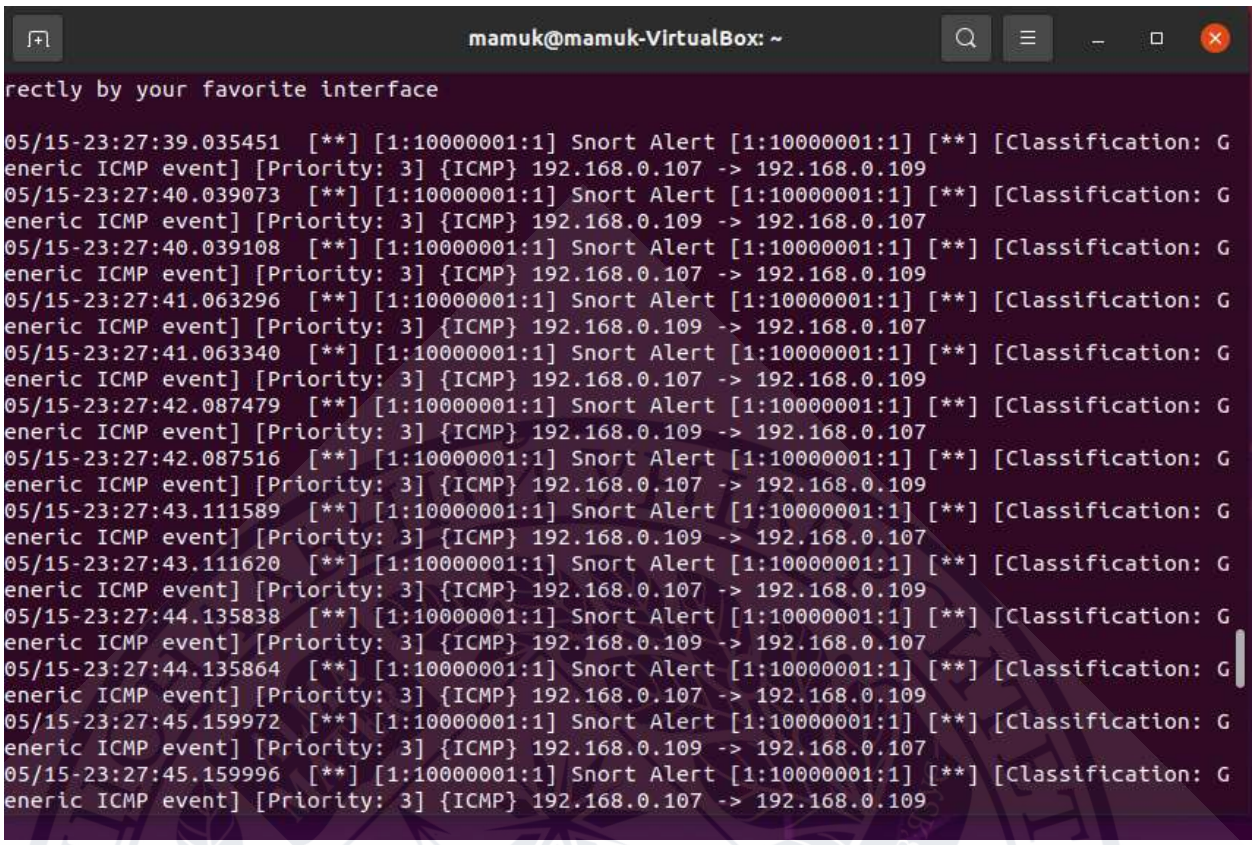

Рисунок 3.3.6 – Результат роботи *Barnyard2*

Для перевірки бази даних використовується наступна команда:

$ *mysql -u snort -p -D snort -e "select count(*) from event"*

Результат приставлений на рисунку 3.3.7.

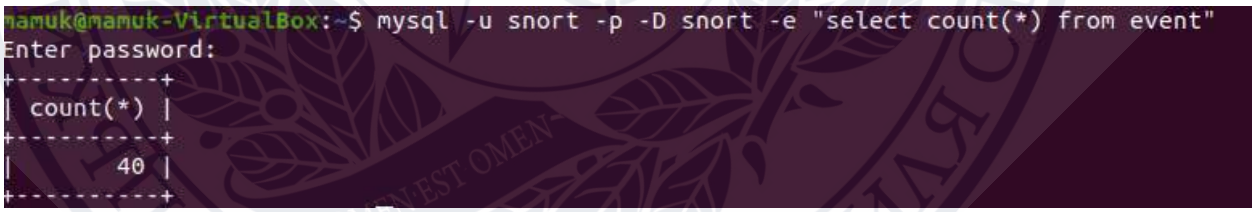

Рисунок 3.3.7 – Результат роботи *MySQL*

Встановлення PulledPork

PulledPork – це скрипт який буде завантажувати та оновлювати правила для Snort. [14]

Встановлення реквізитів та самого скрипта:

$ *sudo apt-get install -y libcrypt-ssleay-perl liblwp-useragent-determined-perl*

$ *cd ~/snort*

$ *wget https://github.com/shirkdog/pulledpork/archive/master.tar.gz -O pulledpork-master.tar.gz*

$ *tar xzvf pulledpork-master.tar.gz*

$ *cd pulledpork-master/*

$ *sudo cp pulledpork.pl /usr/local/bin*

$ *sudo chmod +x /usr/local/bin/pulledpork.pl*

$ *sudo cp etc/*.conf /etc/snort*

Проводиться перевірка що PulledPork працює:

$ */usr/local/bin/pulledpork.pl –V*

Для налаштування *PulledPork* В файлі */etc/snort/pulledpork.conf* вносимо наступні зміни:

- Стрічка 19: вводимо свій *oinkcode* (*rule_url=https://www.snort.org/reg-rules/|snortrules-snapshot.tar.gz*).
- Стрічка 29: необхідно розкоментувати *rule_url=https://rules.emergingthreats.net/|emerging.rules.tar.gz|open-nogpl*.
- Стрічка 74: міняємо шлях до правил на *rule_path=/etc/snort/rules/snort.rules*.
- Стрічка 89: міняємо шлях до наших правил на *local_rules=/etc/snort/rules/local.rules*.
- Стрічка 92: міняємо на *sid_msg=/etc/snort/sid-msg.ma*.
- Стрічка 96: виставляємо другу версію (*sid_msg_version=2*).
- Стрічка 119: вказуємо шлях до конфігу *Snort* (*config_path=/etc/snort/snort.conf*).
- Стрічка 133: вказуємо дистрибутив (*distro=Ubuntu-20-04*).
- Стрічка 141: міняємо шлях до *black_list* на */etc/snort/rules/iplists/black_list.rules*.
- Стрічка 150: змінюємо на *IPRVersion=/etc/snort/rules/iplists*.

Запускаємо *PulledPork* вручну для перевірки його роботи:

$ *sudo /usr/local/bin/pulledpork.pl -c /etc/snort/pulledpork.conf –l*

Результат роботи на рисунку 3.3.8.

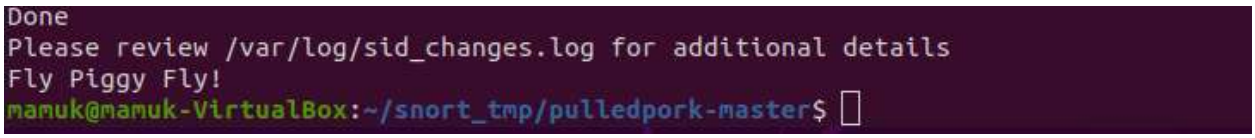

Рисунок 3.3.8 – Результат встановлення *PulledPork*

PulledPork обєднує всі правила в */etc/snort/rules/snort.rules.*

Оскільки файл конфігурації *Snort* був змінений, потрібно перевірити роботу *Snort*. Також це перевірить файл *snort.rules,* створений *PulledPork*: [14]

$ *sudo snort -T -c /etc/snort/snort.conf -i enp0s3*

PulledPork перевірить наявність оновлень і завантажить правила сам. Потрібно просто додати команду, щоб запустити його в планувальнику:

$ *sudo crontab -e*

$ *03 02 * * * /usr/local/bin/pulledpork.pl -c /etc/snort/pulledpork.conf –l*

3.3. Встановлення та налаштування системи виявлення вторгнень *Suricata*

Перед встановленням *Suricata* рекомендується вимкнути служби, що належать *Snort*, оскільки це полегшить навантаження на систему та допоможе уникнути програмних конфліктів. Потім потрібно встановити необхідні реквізити для роботи програми. Деякі з цих реквізитів були встановлені для *Snort*, але тим не менше, краще перевірити їх ще раз: [11, 15]

$ *sudo apt -y install libpcre3 libpcre3-dev build-essential autoconf automake libtool libpcap-dev libnet1-dev libyaml-0-2 libyaml-dev zlib1g zlib1g-dev libmagic-dev libcap-ng-dev libjansson-dev pkg-config libnetfilter-queue-dev geoip-bin geoip-database geoipupdate apt-transport-https*

Для роботи програми необхідно встановити *Java*:

$ *sudo add-apt-repository ppa:linuxuprising/java*

$ *sudo apt update*

$ *sudo apt install oracle-java8-installer*

Після встановлення всіх необхідних реквізитів можна приступити до встановлення самої *Suricata*. Процес встановлення *Suricata* буде дещо іншим,

файли будуть встановлюватися із репозиторіїв, тому не потрібно створювати директорію для *Suricata*.

Додаємо репозиторії Suricata:

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

$ *sudo apt update*

Потім встановлюємо:

$ *sudo apt install suricata*

Необхідно змінити назву мережевих інтерфейсів в файлах */etc/suricata/suricata.yaml* та */etc/default/suricata* з *eth0* на *enp0s3.*

Далі встановлюємо сервіс для оновлення *Suricata* та її правил.

$ *sudo apt install python-pip*

$ *pip install pyyaml*

$ *pip install https://github.com/OISF/suricata-update/archive/master.zip*

Запускаємо оновлення командою:

$ *pip install --pre --upgrade suricata-update*

Далі необхідно оновити джерела правил:

$ *sudo suricata-update update-sources*

Результат оновлення джерел показаний на рисунку 3.4.1.

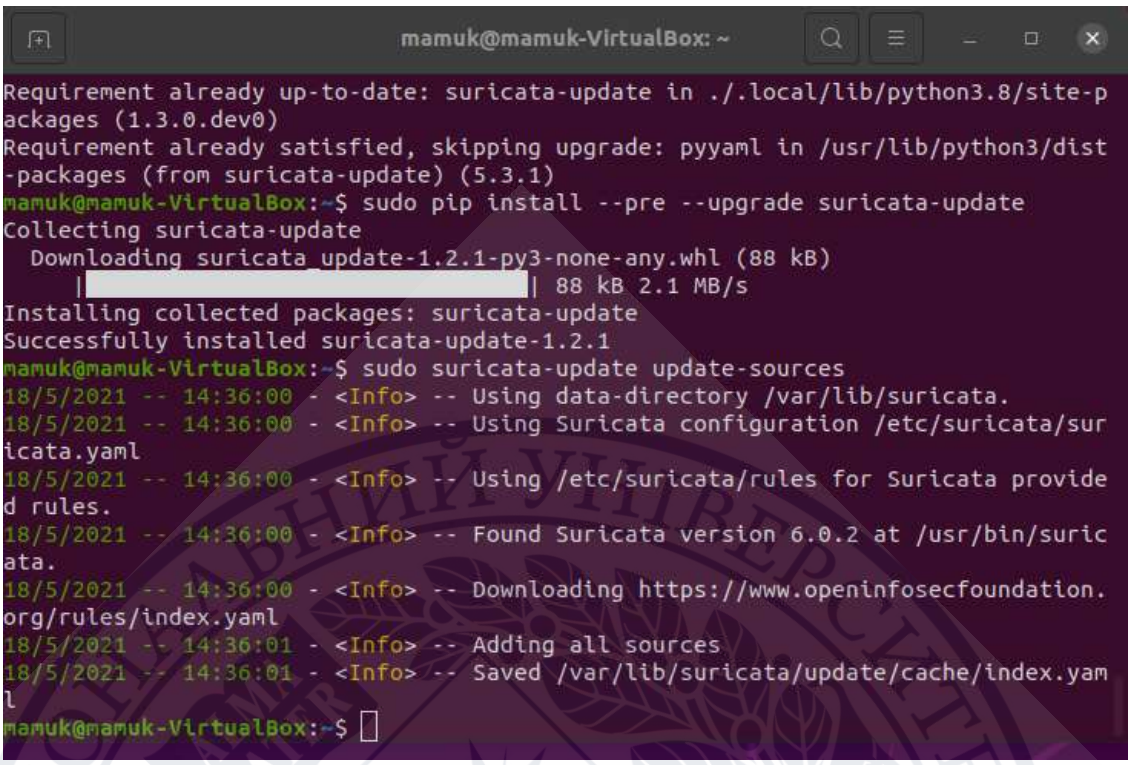

Рисунок 3.4.1 – Результат оновлення джерел правил *Suricata*

Потім оновлюються всі правила:

$ *sudo suricata-update list-sources*

Результат оновлення правил представлений на рисунку 3.4.2.

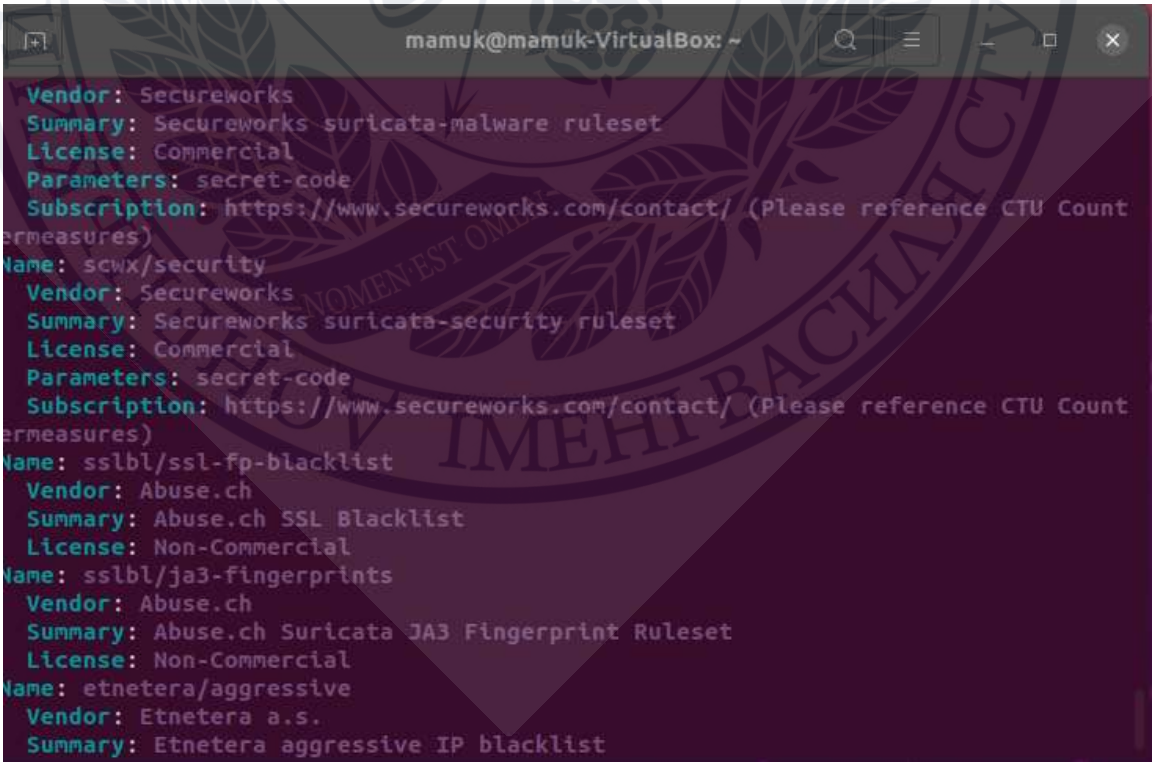

Рисунок 3.4.2 – Результат оновлення правил *Suricata*

Наступним кроком буде ввімкнення всіх безкоштовних джерел з правилами: [11, 15]

$ *sudo suricata-update enable-source ptresearch/attackdetection*

$ *sudo suricata-update enable-source oisf/trafficid*

$ *sudo suricata-update enable-source sslbl/ssl-fp-blacklist*

Результат додавання джерел правил показаний на рисунку 3.4.3.

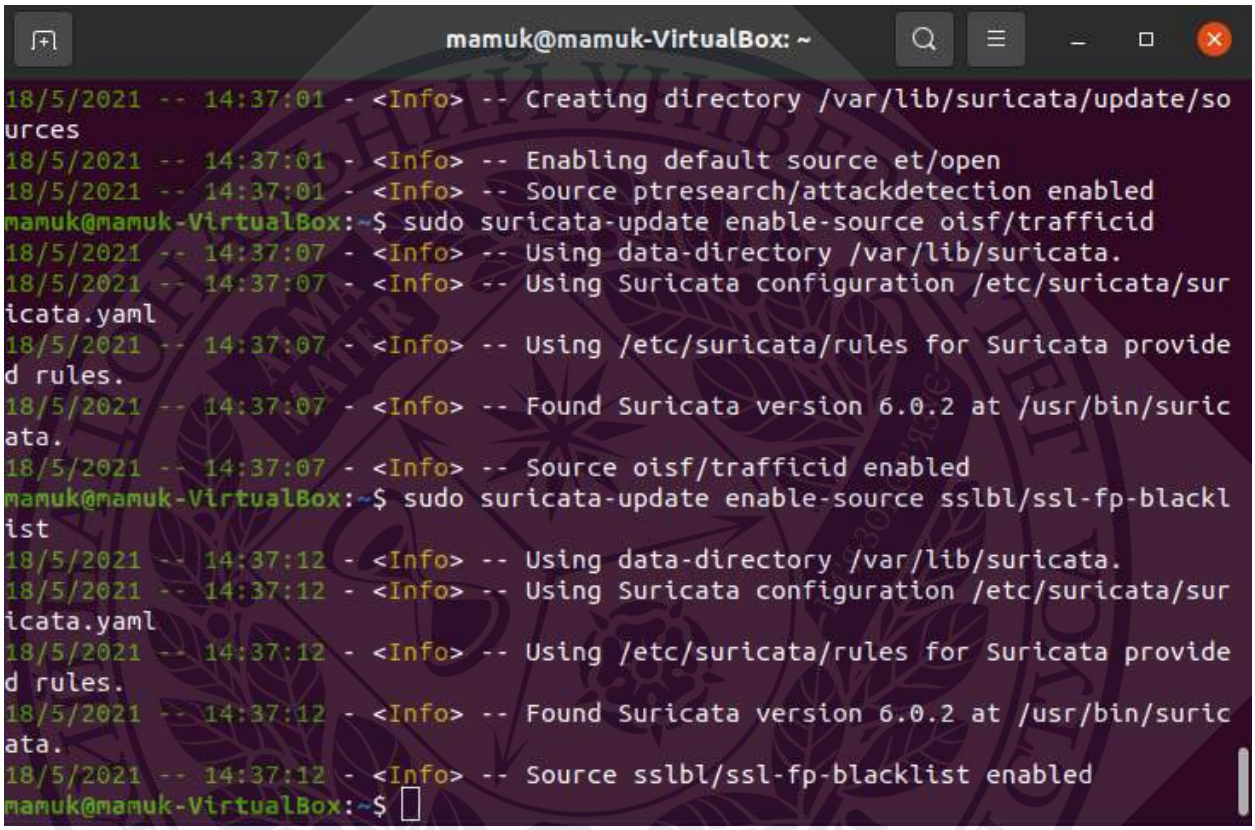

Рисунок 3.4.3 – Результат додавання джерел правил *Suricata*

Запускаємо *Suricata* для перевірки її роботи:

$ *sudo suricata –c /etc/suricata/suricata.yaml –i enp0s3*

Результат роботи *Suricata* показаний на рисунку 3.4.4.

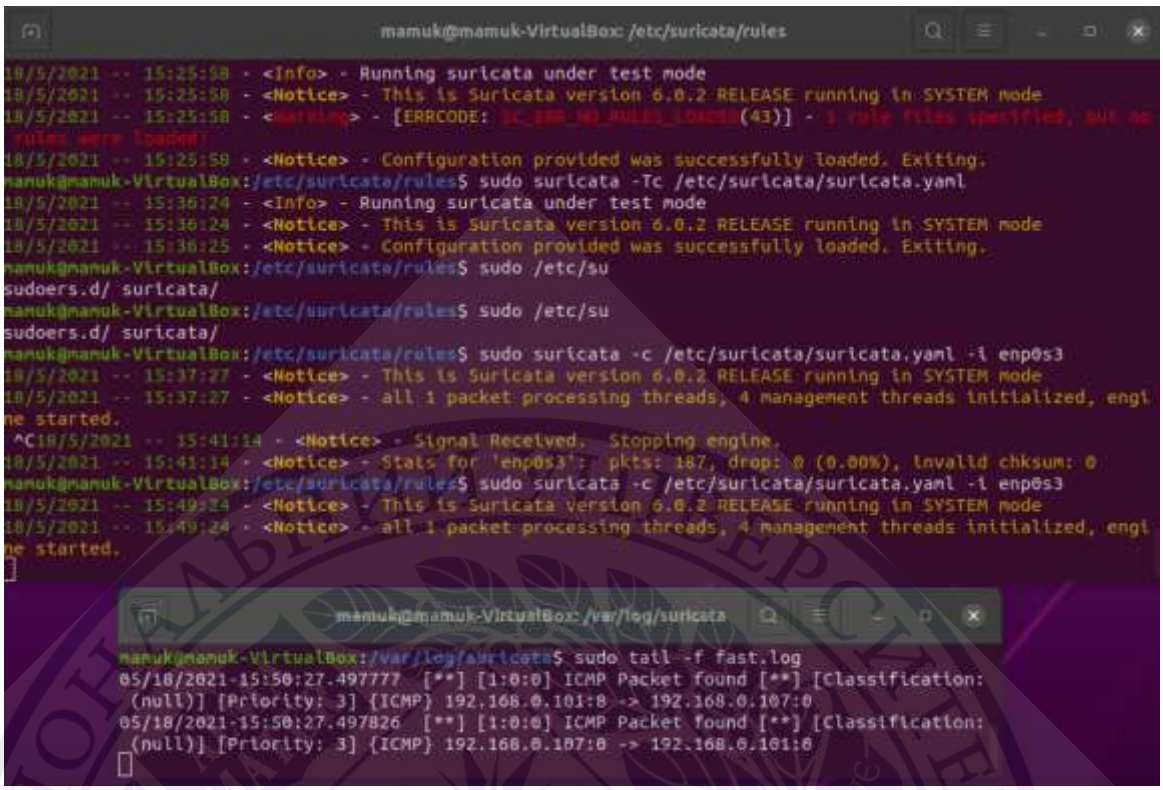

Рисунок 3.4.4 – Результат роботи системи виявлення вторгнень Suricata

Встановлення ELK.

В склад *ELK* входять такі програми як *Elasticsearch, Logstash, Kibana*. Додаємо репозиторій для завантаження програм: [16]

$ *sudo wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add –*

$ *echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list*

Після цього скачуємо та встановлюємо програми:

$ *sudo apt update*

$ *sudo apt-get -y install elasticsearch kibana logstash*

Створюємо службу elasticsearch:

$ *sudo systemctl daemon-reload*

$ *sudo systemctl enable elasticsearch.service*

$ *sudo systemctl start elasticsearch.service*

$ *sudo systemctl status elasticsearch.service*

На рисунку 3.4.5 показаний результат створення служб.

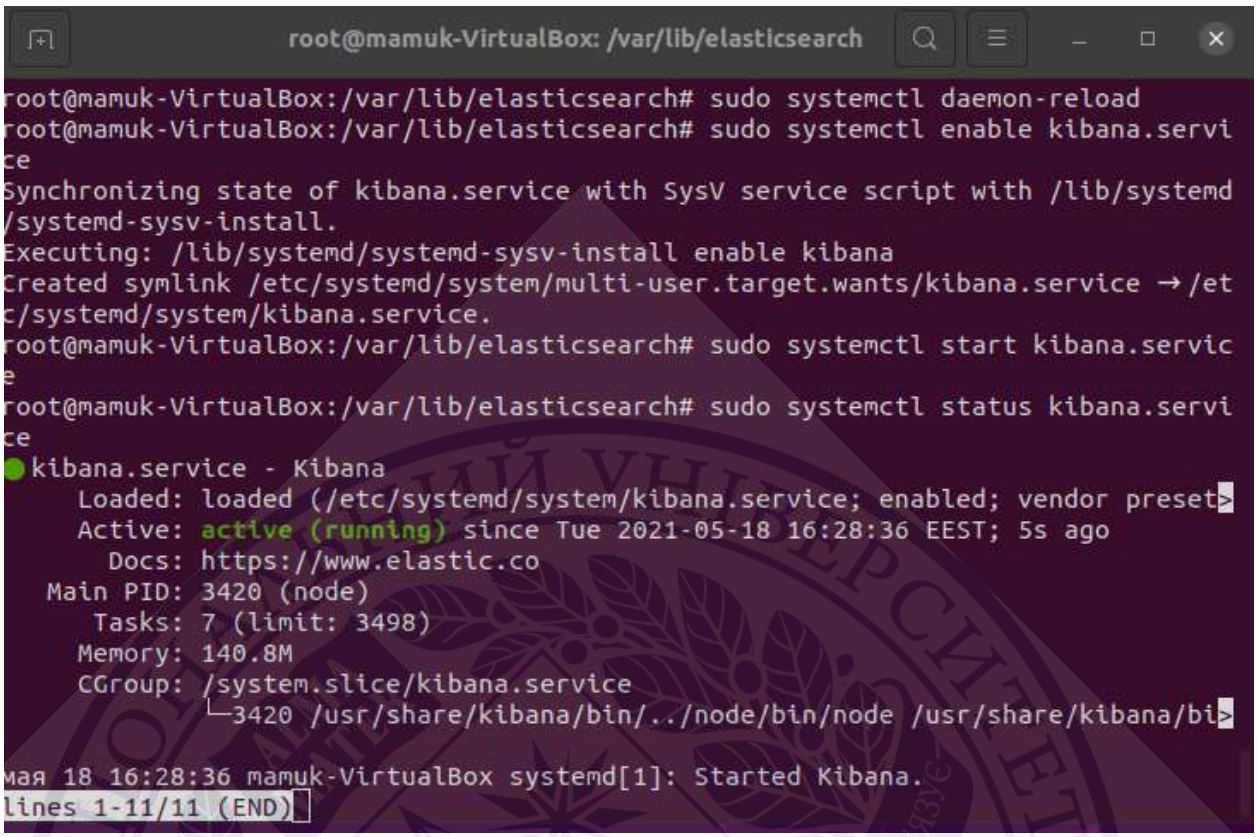

Рисунок 3.4.5 – Створення служби *Elasticsearch*

Наступний етап це налаштування *elasticsearch*. Для цього потрібно змінити файл */var/lib/elasticsearch.* В ньому потрібно змінити наступні параметри: [16]

path.data: /var/lib/elasticsearch/suricata

network.host: 0.0. 0.0

Після цього потрібно перезавантажити службу:

$ *sudo systemctl restart elasticsearch.service*

Щоб перевірити результат можна скористатися командою:

$ *netstat -tulnp | grep 9200*

Далі потрібно створити службу для *Kibana*:

$ *sudo systemctl daemon-reload*

$ *sudo systemctl enable kibana.service*

$ *sudo systemctl start kibana.service*

$ *sudo systemctl status kibana.service*

Щоб перевірити роботу *Kibana* можна скористатися командою:

$ *netstat -tulnp | grep 5601*

Тепер можна відкрити будь-який браузер та перейти по адресі *localhost:5601*.

Результат показаний на рисунку 3.4.6.

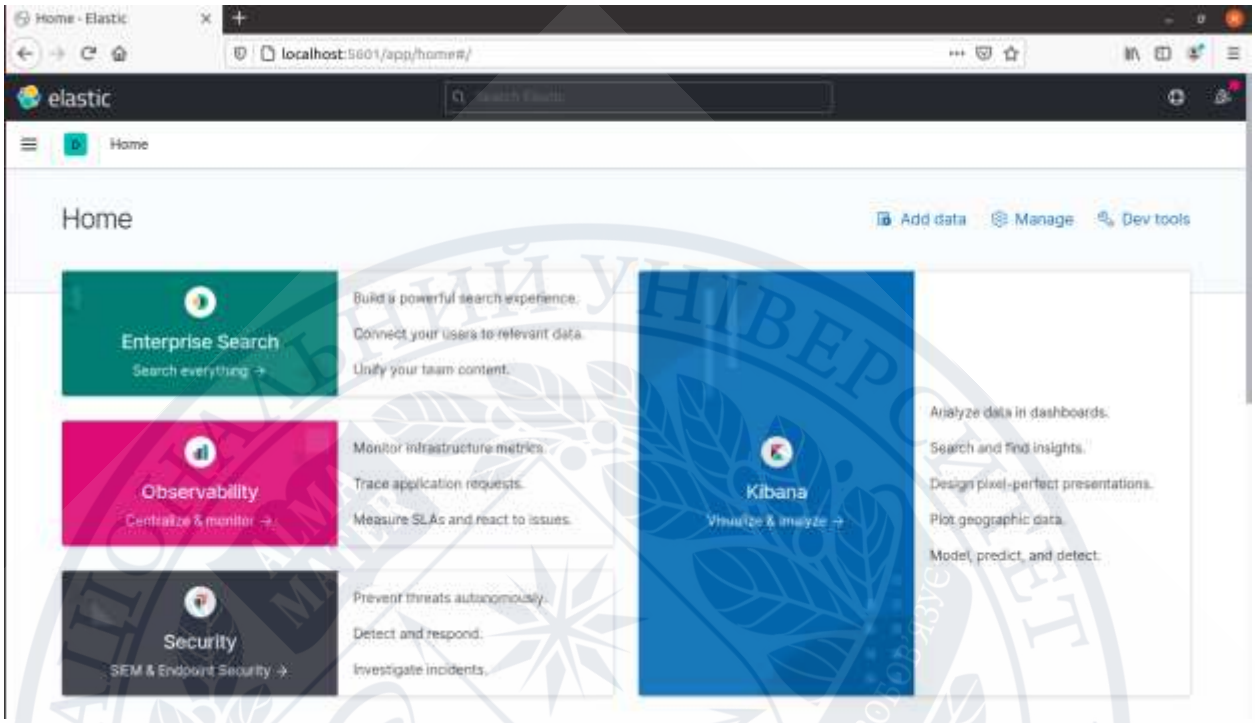

Рисунок 3.4.6 – Завантаження веб додатку *Kibana*

Наступним етапом буде налаштування *logstash*. Він буде передавати інформацію між *elasticsearch* та *kibana*. [16]

Для початку потрібно оновити плагін *mutate*.

$ */usr/share/logstash/bin/logstash-plugin update*

Тепер потрібно налаштувати сам *logstash*. Для його роботи потрібно створити дві конфігураційних файли які будуть описувати введення та виведення інформації.

Файл введення:

$ *sudo gedit /etc/logstash/conf.d/1-input.conf*

Його вміст:

```
Input {
  File {
    Path => ["/var/log/suricata/eve.json"]
    Sincedb_path => ["/var/lib/logstash/"]
```

```
    Codec =>   json
    Type => "SuricataIDPS-logs"
}
}
Filter {
  If [type] == "SuricataIDPS-logs" {
    Date {
      Match => [ "timestamp", "ISO8601" ]
    }
Ruby {
     Code => "if event['event_type'] == 'fileinfo'; event['fileinfo']['type']=event['fileinfo']['magic'].to_s.split(',')[0]; end;"
    }
  }
  If [src_ip]  {
    Geoip {
      Source => "src_ip"
      Target => "geoip"
      Database => "/opt/logstash/vendor/geoip/GeoLiteCity.dat"
      Add_field => [ "[geoip][coordinates]", "%{[geoip][longitude]}" ]
      Add_field => [ "[geoip][coordinates]", "%{[geoip][latitude]}"  ]
    }
    Mutate {
      Convert => [ "[geoip][coordinates]", "float" ]
    }
  }
}
```

Файл виведення:

$ *sudo gedit /etc/logstash/conf.d/1-outputs.conf*

Його вміст:

```
Output {
  Elasticsearch { hosts => ["localhost:9200"] }
  Stdout { codec => rubydebug }
}
```

Після зроблених процедур потрібно повернутися в *Kibana* та вибрати індекс *logstash*-*. Тепер всі дані з *Suricata* будуть відображатися в Kibana.

На рисунку 3.4.7 можна побачити що в *Kibana* відображаються лог-файли з *Suricata* та графік по цим лог-файлам.

Рисунок 3.4.7 – Результат роботи веб додатку *Kibana*

3.4. Порівняння *Snort* та *Suricata*

Протягом багатьох років *Snort* є фактичним стандартом для систем виявлення вторгнень з відкритим кодом. Він поєднує переваги сигнатурного методу, протоколів та методу, заснованого на аномаліях. Через це вона стала широко визнаною, однією з найпопулярніших систем виявлення вторгнень. Порівнюючи системи виявлення вторгнень *Snort* та *Suricata*, можна побачити що вони досить схожі, наприклад, відкритий вихідний код, безкоштовність,

універсальність, кросплатформність. Система виявлення вторгнень *Suricata*, у свою чергу, є новим і менш поширеним продуктом, який з'явився нещодавно, але є досить перспективним. Ця система також базується на сигнатурному методі, але використовує революційні методи. Важливим фактором є те, що, на відміну від *Snort*, режим блокування шкідливого трафіку реалізований з самого початку.

Таблиця 3.5.1 – Порівняння *Snort* та *Suricata* по основним параметрам

Критерій	*Snort*	*Suricata*
Функції системи виявлення вторгнень	*Snort* або *snort_inline* які використовуються з аргументом -Q	Вмикається при компіляції (*--enable nfqueue*)
Правила	*VRT*: Правила *Snort*; Загальні правила	*VRT*: Правила *Snort* Загальні правила
Потоки	Одно поточна	Багато поточна
Складність встановлення	Можливе встановлення з терміналу	
Підтримка *IPv6*	Підтримка реалізована при активації опції (*-enable-ipv6*)	Повна
Документація	Великий обсяг документації	Мало джерел і інтернеті

ВИСНОВОК

В результаті проведеної роботи було проаналізовано ринок систем виявлення вторгнень, а також його характеристики. Проаналізовано необхідність віртуального стенду для аналізу існуючих систем виявлення вторгнень.

У теоретичній частині проведено огляд сучасних технологій систем виявлення вторгнень та архітектури. Також було проведено порівняльний огляд деяких програмних рішень. Для виконання варіантів роботи були обрані такі СВВ як *Snort* та *Suricata*.

В проектній частині роботи був розроблений віртуальний стенд на базі *Ubuntu 20.04* з використанням систем виявлення вторгнень *Snort* та *Suricata*. Крім того, були розроблені інструкції щодо встановлення та налаштування даних СВВ.

З переваг цього віртуального стенду можна виділити його інформативність, багатозадачність та високий потенціал для вдосконалення. При встановленні віртуального стенду допускається заміна елементів, наприклад, для *Suricata* можна встановити додаткові компоненти, які збільшать її інформаційність. Крім того, цей віртуальний стенд є чудовим симулятором для вивчення принципів атак на мережу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Системы и методы обнаружения вторжений: СВВременное состояние и направление СВВершенствования
   URL: http://citforum.ru/security/internet/ids_overview/
2. Безопасность сетей гарантируют системы предотвращения вторжения
   URL:
   http://www.cnews.ru/reviews/free/security2007/articles/networks.shtml
3. Система обнаружения вторжений
   URL:
   https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BE%D0%B1%D0%BD%D0%B0%D1%80%D1%83%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D1%82%D0%BE%D1%80%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9
4. Применение IDS/IPS
   URL: https://xakep.ru/2012/10/29/ids-ips/
5. Классификация IDS
   URL: https://moluch.ru/archive/149/41931/
6. Классификация систем обнаружения атак
   URL: http://citforum.ru/security/internet/ids_overview/
7. Безопасность сетей
   URL: https://www.intuit.ru/studies/courses/102/102/lecture/2995
8. Официальный сайт проекта Snort
   URL: http://www.snort.org/
9. Snort: Часть 1. Введение
   URL: http://csec.ru/snort/Snort-Part-1-Snort-introduction/
10. IDS/IPS-решение Suricata
    https://xserver.a-real.ru/functions/security/sistema-obnarujeniya-vtorjeniy-idsips.php
11. Suricata User Guide

URL: https://suricata.readthedocs.io/en/suricata-4.1.2/

12. Документация программы VMWARE

URL: https://www.vmware.com/ru/products/workstation-pro/

13. Документация программы Oracle VirtualBox

URL:

http://www.oracle.com/us/technologies/virtualization/oraclevm/oraclevmvirtualbox-ds-1655169.pdf

14. Разворачиваем Snort и пишем правила

URL: https://xakep.ru/2018/11/14/snort-rules/

15. Осваиваем сетевую IDS/IPS Suricata

URL: https://xakep.ru/2015/06/28/suricata-ids-ips-197/

16. Опять про IDS или ELK для suricata

URL: https://habr.com/ru/post/280460/

17. Difference between Intrusion Detection System (IDS) & Intrusion Prevention System (IPS) / Asmaa Shaker Ashoor, Prof. Sharad Gore – 2011

18. Study of Snort-Based IDS / S. Chakrabarti, M. Chakraborty, I, Mukhopadhyay - 2010

19. Intrusion Detection and Prevention Systems: Overview of Snort and Suricata / Roman Fekolkin - 2015