

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

«На правах рукопису»

УДК 004.056

«До захисту допущено»

В.о. завідувача кафедри

_____ М.В.Грайворонський

“___” грудня 2019 р.

Магістерська дисертація
на здобуття ступеня магістра

зі спеціальності: 125 Кібербезпека

на тему: _____ *Методи виявлення Honeyrot*

Виконав: студент II курсу, групи ФБ-81мп
(шифр групи)

_____ *Молчанов Євгеній Ігорович* _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник доцент, к.ф.-м.н. Грайворонський М.В.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації
немає запозичень з праць інших авторів без
відповідних посилань.

Студент _____
(підпис)

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
 Кафедра інформаційної безпеки

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою
 Спеціальність (спеціалізація) – 125 Кібербезпека («Системи і технології кібербезпеки»)

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ М.В.Грайворонський
 (підпис)

«__» _____ 2019 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Молчанов Євгеній Ігорович

(прізвище, ім'я, по батькові)

1. Тема дисертації Методи виявлення Honeypot

науковий керівник дисертації Грайворонський Микола Владленович,
доцент, к.ф.-м.н.

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «15» 11 2019 р. № 3927

2. Термін подання студентом дисертації 10.12.2019 р.

3. Об'єкт дослідження Honeypot

4. Вихідні дані програма виявлення Honeypot Cowrie, критерії стійкості до розпізнавання Honeypot. 5.

Перелік завдань, які потрібно розробити 1. Ознайомитись з концепцією та різними реалізаціями Honeypot. 2. Розглянути та проаналізувати вже існуючі рішення для можливості вивчення Honeypot. 3. Обрати та розгорнути три різні реалізації Honeypot терміном не менше 5 днів. 4. Проаналізувати отримані результати, а саме: кількість атак, кількість унікальних IP-адрес, кількість завантажених шкідливих файлів та кількість користувачів які намагалися встановити, що вони знаходяться у Honeypot. 5. Обрати серед розгорнутих Honeypot ту приману, на яку було спрямовано найбільшу кількість атак та розробити програму для її виявлення. 6. Проаналізувати роль застарілого обладнання у якості ознаки

присутності у Honeypot. 7. Розробити стратегію стартап-проекту на основі виконаної роботи задля вирішення проблеми виявлення Honeypot.

6. Орієнтовний перелік ілюстративного матеріалу 1.1. Приклад використання Honeypot. 2.1.1. Схема розгорнутих Honeypot. 2.2.1. Архітектура HoneySMB. 2.3.1 Архітектура HoneyWEB-SQL. 2.5.1 Архітектура ELK Stack. 3.1.1. SMB комунікація. 3.1.2. Сценарій атаки на SMB. 3.3.1. Кількість атак у день після розгортання.

7. Орієнтовний перелік публікацій

Результати дослідження були подані до публікації у журналі “Theoretical and applied cybersecurity”.

8. Дата видачі завдання 01.05.2019

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	01.05.2019 – 01.06.2019	
2	Огляд існуючих рішень	01.06.2019 – 01.07.2019	
3	Пошук інформації	01.07.2019 – 01.08.2019	
4	Практичне дослідження можливості виявлення Honeypot	01.08.2019 - 01.09.2019	
5	Аналіз отриманих результатів	01.10.2019 – 01.11.2019	
6	Створення програмного продукту	01.11.2019 — 20.11.2019	
7	Створення стартап-проекту	20.11.2019 — 05.12.2019	
8	Подання дисертації на передзахист	06.12.2019	

Студент

(підпис)

(ініціали, прізвище)

Науковий керівник дисертації

(підпис)

(ініціали, прізвище)

РЕФЕРАТ

НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ

виконану на тему “Методи виявлення Honeypot”

студентом групи ФБ-81мп Молчановим Євгенієм Ігоровичем

Робота виконана на 95 сторінках, містить 11 ілюстрацію, 28 таблиць. При підготовці використовувалась література з 8 джерел.

Актуальність. Зловмисники стають все більш обережними під час проведення атак та дослідженні потенційних цілей, намагаючись виявити Honeypot або зламати його. У свою чергу, це знижує ефективність цього інструменту зі збору та аналізу дій зловмисників. Саме тому, є необхідність у приверненні більшої уваги до методів виявлення та стійкості до розпізнавання Honeypot.

Мета. Мета даної роботи полягає у дослідженні сучасних методів виявлення найпоширеніших реалізацій Honeypot.

Завдання. Для успішного досягнення мети даної роботи необхідно виконати наступні завдання:

- Ознайомитись з концепцією та різними реалізаціями Honeypot;
- Розглянути та проаналізувати вже існуючі рішення до можливості виявлення Honeypot;
- Обрати та розгорнути три різні реалізації Honeypot терміном не менше 5 днів;
- Проаналізувати отримані результати а саме: кількість атак, кількість унікальних IP-адрес, кількість завантажених шкідливих файлів та кількість користувачів які намагалися встановити що вони знаходяться у Honeypot;

- Обрати серед розгорнутих Honeypot, ту приманку, на яку було спрямовано найбільшу кількість атак та розробити програму для її виявлення;
- Проаналізувати роль застарілого обладнання у якості ознаки присутності у Honeypot;
- Розробити стратегію стартап-проекту на основі виконаної роботи задля вирішення проблеми виявлення Honeypot;

Об'єкт дослідження. Об'єктом дослідження є Honeypot.

Предмет дослідження. Предметом дослідження є методи виявлення Honeypot.

Наукова новизна. Науковою новизною роботи є дослідження методів виявлення Honeypot, можливість їх автоматизації та розробка критеріїв стійкості до розпізнавання.

Практична цінність. Практичною цінністю роботи є аналіз, створення програми для автоматизації можливості виявлення одного з найпоширеніших реалізацій Honeypot та розробка критеріїв стійкості до розпізнавання.

Ключові слова. Honeypot, Detection, HoneyDetection, Bait, Attacks, HoneySMB, HoneyWEB-SQL, Cowrie, Resistance criteria

ABSTRACT

ON MASTER'S THESIS

on topic "Honeypot detection methods" student Molchanov Yevhenii

Master's thesis is made on 95 pages, contains 11 illustrations, 28 tables. There were 8 sources used in the preparation.

Topicality. Attackers are becoming more cautious when conducting attacks, and exploring potential targets, trying to identify or break a Honeypot. In turn, this reduces the effectiveness of this tool in collecting and analyzing the actions of intruders. That is why there is a need to pay more attention to Honeypot protection and detection methods.

Purpose. The purpose of this work is to investigate current methods of detecting the most common Honeypot implementations.

Tasks. To successfully achieve the goal of this work, I need to perform the following tasks:

- Familiarize yourself with the concept and various Honeypot implementations;
- Review and analyze pre-existing solutions to Honeypot detection;
- Select and deploy three different Honeypot implementations for at least five days;
- Analyze the results, namely: the number of attacks, the number of unique IPs, the number of malicious files downloaded, and the number of users trying to determine that they are in Honeypot;
- Choose among the deployed Honeypot, the one that attracted the most significant number of attacks and develops a program to detect it;
- Analyze the role of legacy equipment as a Honeypot presence feature;

- Develop a startup project strategy based on work done to solve the Honeypot detection problem;

Object of research. The object of research is Honeypot.

Subject of research. The subject of the research is the Honeypot detection methods.

Scientific novelty. The scientific novelty is the research of Honeypot detection methods, the possibility of their automation, and the development of criteria for resistance to recognition.

Practical value of research. The practical value of the work is analysis, the creation of a program to automate the ability to identify one of the most common Honeypot implementations, and the development of persistence criteria for Honeypot recognizing.

Publications.

Keywords. Honeypot, Detection, HoneyDetection, Bait, Attacks, HoneySMB, HoneyWEB-SQL, Cowrie, Resistance criteria

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів.....	2
Вступ.....	4
1 Огляд існуючих рішень.....	8
1.1 Історія і значення honeypot.....	8
1.2 Виявлення honeypot UML.....	13
1.3 Виявлення і атака на VMWare honeypot.....	16
1.4 Виявлення середовища chroot.....	18
1.5 Виявлення honeyd honeypot.....	19
1.6 Загальні проблеми з honeypot.....	20
Висновок до розділу 1.....	23
2 Відомості про метод розв'язання.....	24
2.1 Порівняльний аналіз Honeypot з конкурентними рішеннями	24
2.2 Архітектура розгорнутих приманок.....	28
2.3 Розгортання HoneySMB.....	29
2.4 Розгортання HoneyWEB-SQL.....	31
2.5 Розгортання Cowrie.....	33
2.6 Розгортання ELK Stack.....	34
2.7.1 Аналіз результатів отриманих з HoneySMB.....	36
2.7.2 Аналіз результатів отриманих з HoneyWEB-SQL.....	40
2.7.3 Аналіз результатів отриманих з Cowrie.....	42
Висновок до розділу 2.....	43
3 Методи виявлення Honeypot.....	44
3.1 Узагальнення методів виявлення Honeypot.....	44
3.2 Виявлення honeypot на базі застарілого обладнання.....	45
3.3 Розробка програми для виявлення Cowrie honeypot.....	47
3.4 Критерії стійкості honeypot до розпізнавання.....	51

Висновки до розділу 3.....	51
4 Розробка стартап-проекту “Автоматизація виявлення Honeypot”.....	55
4.1 Опис ідеї проекту.....	55
4.2 Технологічний аудит ідеї проекту.....	57
4.3 Аналіз ринкових можливостей стартап-проекту.....	58
4.4 Розроблення ринкової можливості стартап-проекту.....	68
4.5 Розроблення ринкової стратегії стартап-проекту.....	73
4.6 Розроблення маркетингової програми стартап-проекту.....	77
Висновок до розділу 4.....	78
Висновок.....	80
Перелік джерел посилань.....	81
Додаток А.....	82

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Honeyrot - приманка у цифровому вигляді виконана у формі ресурсу, сервісу тощо.

VMWare - програма для віртуалізації програмного середовища.

Chroot - опція зміни кореневого каталогу в Unix подібних системах.

UML - режим користувача Linux.

Shodan - пошукова система яка дозволяє знаходити пристрої різного типу, підключені до мережі інтернет

ВСТУП

Оскільки світ стає все більш інформаційним і орієнтованим на мережеву взаємодію, виникає необхідність у забезпеченні безпеки. У минулому практикуючим фахівцям у галузі кібербезпеки доводилося чекати, поки відбудеться атака, перш ніж вони зможуть її проаналізувати, а потім віднайти методи можливого захисту або способи пом'якшення шкоди від неї. Важко передбачити, який вектор ініціалізації, приховані канали або режим роботи будуть використовувати зловмисники у спробі зламати або відключити систему. Щоб спробувати дізнатися, як працюють зловмисники і які методи вони використовують, була розроблена технологія, яка називається «Honeyrot». По суті, honeyrot - це система, яка імітує слабку або вразливу систему, намагаючись привернути увагу потенційного зловмисника, щоб спровокувати його проникнути або зламати машину, поки honeyrot реєструє всі події, що відбуваються. Під час підготовки до виконання поставленого завдання мною були опрацьовані матеріали в яких описані способи розгортання honeyrot, слабкі сторони, які є у кожного типу honeyrot, і методи, які зловмисники можуть використати, щоб визначити, чи дійсно вони знаходяться всередині honeyrot.

З розвитком технології honeypots співтовариство та бізнес знаходять різні способи його використання. Перший з цих методів відомий як призначений для користувача режим Linux або UML. UML - це ядро Linux, яке можна запустити з іншої системи Linux. Таким чином, користувач має можливість пересилати будь-які пакети або мережеву активність в UML, щоб дати людині, що намагається встановити з'єднання, відчуття, що він насправді знаходиться всередині звичайної системи Linux. В системі UML користувач може налаштувати будь-яку програму або службу для роботи так само, як на звичайній машині Linux. Коли зловмисники намагаються підключитися до певної служби, наприклад ssh,

вони отримують хороше уявлення про систему, тому що фактично це буде та ж сама служба. Перевага UML полягає в тому, що є можливість вести запис ТТҮ. Більшість honeypots робить запис перехоплюючи пакети що працює добре, поки зловмисник не вирішить підключитися через ssh. Перевага ведення журналу ТТҮ полягає у можливості логування кожної дії зловмисника. Це дуже допомагає проаналізувати та зрозуміти як саме сталася атака.

VMWare - це ще один інструмент, який стає все більш популярним для створення honeypot. VMWare була створена для роботи в якості віртуальної машини, яка дає можливість користувачам запускати іншу операційну систему в поточному робочому середовищі. Віртуальна машина VMWare емулює обладнання, надаючи набір драйверів для віртуальних пристроїв. Оскільки ці драйвери однакові на всіх віртуальних машинах VMWare, образи легко переносяться з одного хоста на інший. Після присвоєння віртуальній машині VMware IP-адреси з нею можна зв'язатися по мережі та досить легко змоделювати конкретну схему хоста і мережі. Також однозначною перевагою є можливість перенаправлення трафіку через honeypot.

Ще одним популярним рішенням є приманка яка має назву honeyd. Honeyd - це демон, який був написаний для платформ UNIX, але тепер існує і для Windows. Він призначений для емуляції декількох хостів і мереж на одному комп'ютері. Це досягається шляхом повернення пакетів. Таким чином, якщо хтось виконає сканування мережі, щоб спробувати визначити типи хостів, проаналізувавши перехоплені пакети ми зможемо це констатувати. Деякі служби, наприклад telnet на маршрутизаторі Cisco, може емулюватися шляхом встановлення додаткових модулів або написання сценаріїв за допомогою обраної мови програмування. Сценарії дозволять зловмисникам підключитися до хоста через цей сервіс на відповідному порті і створять враження, що він приєднується до звичайної машини.

Honeyd працює, перехоплюючи пакети, спрямовані на певні IP-адреси і порти. У конфігураційному файлі зберігаються основні налаштування. На одному хості honeyd є 65536 IP-адрес які можуть бути використані для емуляції мережі. Враховуючи перераховані переваги honeyd є дуже гнучким рішенням для використання у якості приманки.

Іншим методом створення приманки є використання середовища chroot або jail. Це рішення схоже на приманки створені за допомогою засобів віртуалізації. Відмінність полягає у тому що це рішення запускається на тому ж хості і ядрі, що і основна операційна система. Середовище Chroot в основному обмежує користувача заздалегідь визначеною областю системи. Це дозволяє створити невелике емітоване середовище з якого зломисник не має змоги зашкодити основній системі. Наприклад у вас створено середовище honeypot в директорії /tmp/honeypot, ви можете встановити для користувачів chroot значення /tmp/honeypot, і коли вони будуть потрапляти до систему, це виглядатиме ніби вони знаходяться у кореневій директорії. Звідси ви можете налаштувати свій імітований сервер та почати логувати дії зломисників щоб після віднайти методи та цілі які мав зломисник. Для того щоб система здавалася більш звичайною в ній вже повинні бути створенні директорії і файли. Наприклад це може включати каталоги /dev і /proc, а також стандартні файли операційної системи, які існують в директоріях /usr і /etc. З урахуванням вищесказаного можна було змодельовати різні версії або дистрибутиви Linux, помістивши специфічні для ОС файли в chroot.

Схоже що honeypot - це компетентний інструмент, який допомагає спеціалістам з кібербезпеки розкрити методи які були використані зломисниками та розробити засоби захисту. Тим не менш зломисники все частіше намагаються встановити або зламати Honeypot.

Honeypot цікавий зловмисникам, тим що більшість з них не хочуть, щоб їхні методи стали відомі. Якщо хакер виявить вразливість нульового дня на сервері або в програмному забезпеченні він може проексплуатувати, продати або звернутися до компанії за винагородою. Якщо зловмисник віднайде та проексплуатує цю вразливість на приманці то його знахідка, швидше за все, стане загальнодоступною, що в свою чергу знизить цінність або анулює її взагалі. Саме через це зловмисниками були розроблені методи, які допомагають їм визначити, чи знаходяться всередині honeypot та способи їх зламу.

Висвітлення даного питання має відображення у працях Corey, J. *Advanced Honey Pot Identification and Exploitation* (2005), серії книжок L. Holz *Defeating Honeypots: System and Networks issues* (2004). Тим не менш, беручи до уваги, що пройшло більше десятиліття з моменту публікацій останньої всесвітньо визнаної роботи та того факту, що в них не було представлено узагальнених принципів та критеріїв за якими можна було б оцінити ступінь важкості розпізнавання тієї чи іншої реалізації Honeypot. Саме тому висвітлення цього питання є актуальним та значущим для галузі в цілому.

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Історія і значення Honeypot

Вперше ідея створення Honeypot з'явилася у двох публікаціях від 1991 року: «The Cuckoos Egg» та «An Evening with Breford». У першій публікації «The Cuckoos Egg» Кліффорда Столла описувався його досвід з відстеження кіберзлочинця, який був впроваджений та знаходився у середині його корпорації намагаючись поцупити секретні дані.

В іншій публікації «An Evening with Breford» Білла Чевіка - про проникнення кібер злочинця до системи через пастки, які Білл та його колеги використовували щоб спіймати його.

Перший тип honeypot був розроблений в 1997 році та мав назву Desceptive Toolkit його основною ідеєю було використання обману для атаки у відповідь. У 1998 році з'явився перший комерційний honeypot який мав назву Cybercor Sting, з 2002 року його можна було вільно поширювати та використовувати. З того часу технологія honeypot значно покращилася і багато експертів вважають, що це лише початок. У 2005 році, у філіппінах стартувала державна ініціатива з використання Honeypot, який був запущений з метою сприяння безпеці комп'ютерів на Філіппінах.

Отже, Honeypot створений для залучення та моніторингу дій користувачів, які намагаються проникнути в комп'ютерні системи інших осіб. Honeypot - це пастка; приманка у цифровому вигляді. Це комп'ютер, віртуальні або мережеві ресурси, які на перший погляд здаються звичайними але використовуються для того щоб того щоб заманити зловмисника.

Більшість honeypots встановлюються всередині мережевих брандмауерів і є засобом контролю і відстеження зловмисників. Honeypots і брандмауери працюють в зворотньому напрямку, оскільки honeypots дозволяють весь вхідний трафік, але блокує весь вихідний трафік у той час як брандмауер дозволяє вихідний трафік та блокує більшість вхідного трафіку відповідно до зазначених правил. Більшість honeypots.

Honeypots - це унікальний інструмент для вивчення поведінки зловмисників але його потрібно використовувати обережно, оскільки існує ймовірність злому. Зловмисник може використовувати honeypot для прориву в основну систему. Отже, Honeypot повинен бути максимально відгороджений від діючих систем.

Взагалі, ідея створення honeypot полягає в тому, щоб встановити систему "приманку", яка має незахищену операційну систему, або систему, яка має декілька вразливостей, використавши які можна з легкістю отримати доступ до її ресурсів.

Honeypot ресурс не має реального використання. Іншими словами, звичайні користувачі, скоріш за все ніколи не підключається до нього. Він налаштований таким чином, щоб привернути увагу зловмисників які будуть атакувати його. Виходячи з цього можемо констатувати, що звичайний користувач, котрий підключається до ресурсу honeypot, у 99% випадків є шкідливим.

Концепція Honeypots полягає в тому, щоб відловлювати шкідливу мережеву активність на підготовленому ресурсі котрий використовується в якості приманки. Цінні скомпрометовані дані збираються за допомогою програмного забезпечення. Ця інформація є скоріше інструментом спостереження і раннього попередження, яка також є допоміжним засобом для комп'ютерної та мережевої експертизи. Зазвичай зловмисник спочатку повинен виявити Honeypot і

спробувати проникнути в нього. Потім вже виходячи з типу і призначення Honeypot можна констатувати що саме зловмисник може виконати.

Дві основні причини через які розгортають honeypots:

- 1) Щоб дізнатися, як зловмисники досліджують та намагаються отримати доступ до ваших систем. Отримати уявлення про методи атак які були використані зловмисниками, щоб у свою чергу покращити захист системи.
- 2) Збір інформації, необхідної для допомоги у затриманні або переслідуванні зловмисників.

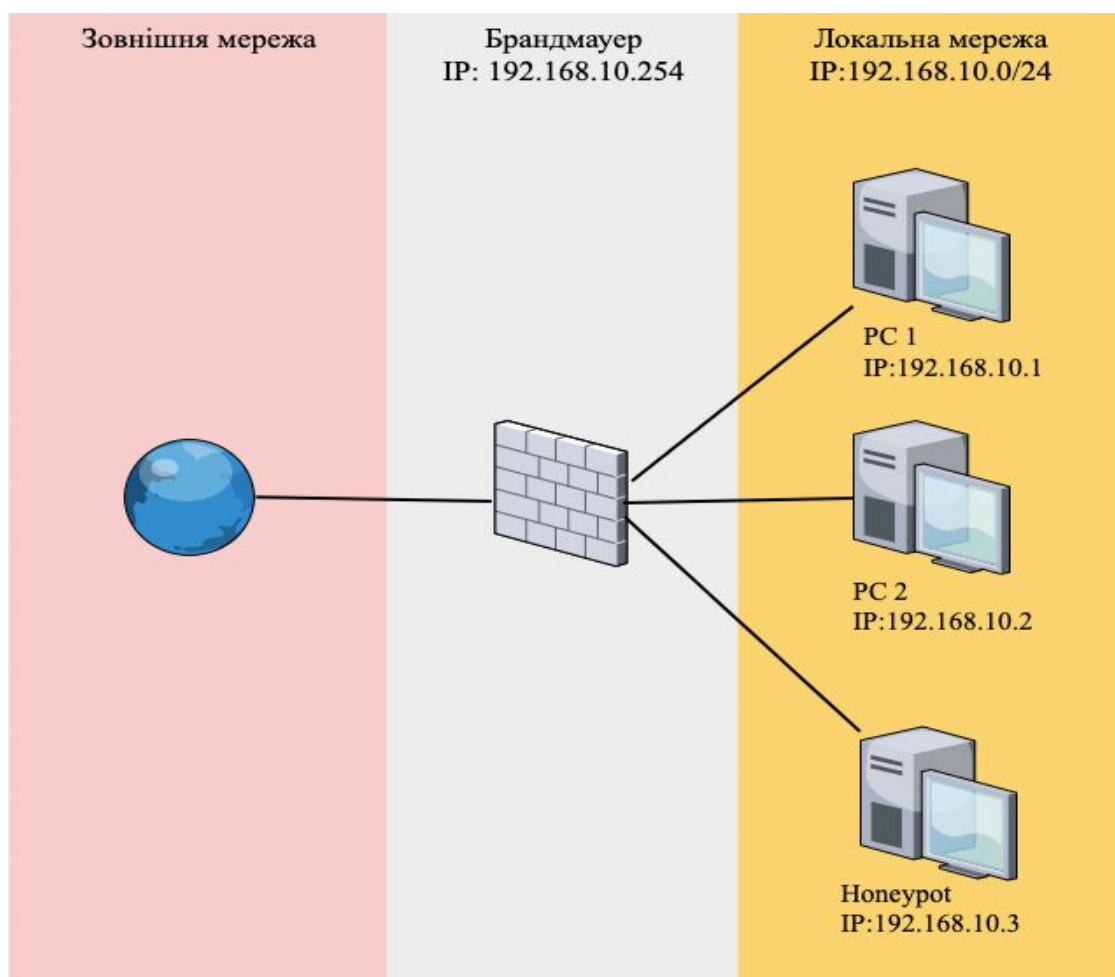


Рисунок 1.1, Приклад використання Honeypot

Потрібно зазначити що honeypot не має бути зареєстрований на жодному з серверів імен або будь-яких інших системах, задля приховування його існування. Це важливо, тому що тільки в правильно налаштованій мережі можна припустити, що майже кожен пакет, відправлений в Honeypot з метою атаки. При помилковому налаштуванні, кількість помилкових попереджень стрімко зростає і це, може призвести до відмови в обслуговуванні.

Цінність отриманої інформації за допомогою Honeypot може різнитись в залежності від його використання. Цінність самого Honeypot, у свою чергу, залежить від того, наскільки отримана інформація допоможе запобігати наступним атакам.

Способи які допоможуть захистити організацію за допомогою Honeypot:

1) Перший спосіб спрямований проти автоматичних атак, таких як хробаки та авто-роутери. Ці атаки засновані на інструментах, які випадковим чином сканують цілі мережі в пошуках вразливостей. Якщо будуть виявленні вразливості, ці автоматизовані інструменти будуть намагатися поексплуатувати вразливість і проникнути в систему. Одним із способів захисту від таких атак за допомогою Honeypot є уповільнення їх сканування та навіть зупинка. Ці рішення мають назву «sticky honeypots», вони відстежують не використаний простір IP. При виявленні сканування ці приманки взаємодіють з атакуючим і сповільнюють його, використовуючи різні прийоми TCP, такі як «Windows size of zero», переводячи зловмисника у режим очікування. Це гарний спосіб для уповільнення або запобігання поширенню черв'яка, який проник в вашу внутрішню організацію. Одним з таких прикладів «sticky honeypots» є LaBrea Tarpit. Так звані «липкі» приманки найчастіше є рішеннями з низьким рівнем взаємодії, оскільки вони сповільнюють атакуючого до мінімуму.

2) Другий спосіб, який допоможе захистити організацію за допомогою Honeypot є виявлення. Виявлення є критично важливим і має на меті виявлення атаки, збоїв або іншого що є критично важливим для організації. Незалежно від того, наскільки захищеною є організація, перед нею завжди постають певні ризики та виклики. Виявивши атаки, ви можете швидко реагувати, зупиняючи або пом'якшуючі шкоду, яку вони завдають. Традиційно виявлення є надзвичайно складним завданням. Такі технології, як IDS та системи журналів, виявилися неефективними з певних причин. Вони генерують занадто багато даних, мають великий відсоток помилкових спрацьовувань, не мають можливості виявлення нових атак і не мають можливості працювати з шифрованим трафіком або IPv6 середовищі. У порівнянні Honeypot є більш гнучким засобом і вирішує багато з проблем традиційних способів виявлення.

3) Третім способом, який допоможе захистити організацію за допомогою Honeypot є реагування на проникнення. Найчастіше це один з найбільших викликів, з якими стикається організація. Зазвичай у спеціалістів з кібербезпеки дуже мало інформації про те, хто є зловмисником, яким чином він потрапив до системи та чи скільки він встиг завдати шкоди. У таких ситуаціях інформація про діяльність зловмисника є критичною.

Існує дві основні проблеми, що ускладнюють можливу відповідь на проникнення. По-перше, найчастіше скомпрометовані системи не можуть бути переведені до автономного режиму для аналізу. Виробничі системи, такі як поштовий сервер організації, настільки критичні, що, навіть якщо вони були скомпрометовані, фахівці з кібербезпеки можуть не мати можливості вимкнути систему і провести належний аналіз інциденту. Замість цього вони обмежені в аналізі діючої системи, продовжуючи при цьому надавати послуги. Через це не завжди можливий детальний аналіз того що трапилося і якої шкоди завдав зловмисник. По-друге якщо навіть система відключена, є можливість того що

данні настільки забрудненні, що дуже важко визначити, що саме зробив зловмисник. Під забрудненням даних я маю на увазі велику активність (вхід користувача в систему, перегляд електронної облікових записів, файлів, запис в бази даних і тд.), Може бути важко відрізнити, що є звичайною, повсякденною діяльністю, а що ні і хто саме проводим атаку. Honeypot може допомогти вирішити обидві проблеми. Honeypot є відмінним інструментом для аналізу інцидентів, оскільки приманку можна швидко відключити для проведення повного детального аналізу, ніяким чином не впливаючи на повсякденну діяльність системи. Крім того, honeypot поглинає лише несанкціонований або шкідливий вплив. Це значно спрощує аналіз зламаних honeypot, у порівнянні зі зламаними виробничими системами, оскільки будь-які дані, отримані з honeypot, швидше за все, пов'язані зі зловмисником. У такому ключі цінність, яку несе honeypot, - це швидке надання організаціям повної інформації, необхідної їм для швидкого і ефективного реагування на інциденти.

1.2 Виявлення Honeypot UML

Режим користувача у Linux має досить багато проблем, які можуть бути досить легко виявлені зловмисником, який добре знає, що саме він шукає. По-перше UML не використовує справжні жорсткі диски для зберігання даних. Замість цього він використовує віртуальні пристрої, які в свою чергу вказують на існуючі частини файлової системи, котрі містять образи дисків. Ці образи монтуються в системі UML як `/dev/ubd*`. Найпростіший спосіб перевірити це - виконати команду «`df -k`» або «`mount`», щоб дізнатися, що саме і де монтується. У більш пізніх випусках UML були розроблені модулі, щоб спробувати приховати пристрої `/dev/ubd*` і зробити їх схожими на справжні диски. Для доступу до цих модулів користувач повинен запустити UML з включеними опціями `fake idea` та `fakehd`. Веж це має обмеження які полягають у тому що основний номер, що

ідентифікує пристрої `/dev/ubd*`, не збігається з тим, який використовується для стандартних систем IDE або SCSI. Це число 98(0x62).

Ще один швидкий спосіб дізнатися, чи перебуваєте ви всередині UML замість реальної системи, - поглянути на `/proc tree`. Тут є кілька аномалій, які досить очевидні для обізнаного зловмисника.

По-перше виконаємо команду `cpuinfo`:

```
cat /proc/cpuinfo
```

```
processor: 0  
  
vendor_id: User  
  
model_name: UML  
  
mode: tt  
  
[...]
```

Тут є дві очевидні проблеми які полягають у тому, що назва моделі вказана як UML, а поточний режим - Tracing Thread.

Виконаймо команду `egrep`:

```
egrep "uml|honey" /proc/ksysms
```

```
a02eb408 uml_physmem  
  
a02ed688 honeypot
```

Файл `ksysms` відображає виділення пам'яті для `honeypot`.

Далі перевіримо пристрої та виконаємо команду:

```
cat /proc/devices
```

[...]

Block devices:

[...]

60 cow

90 ubd

З цього бачимо що диски позначені як ubd, як вони згадуються в /dev/ перед запуском fakehd.

Нарешті перевіримо файлову систему і виконаємо команду:

```
cat /proc/filesystems
```

[...]

```
nodev hostfs
```

З цього бачимо що запущена файлова система hostfs, що у свою чергу вимагає зворотнього зв'язку з файловою системою hosts це є ознакою віртуального середовища. Був розроблений інструмент під назвою «hprfs» (файлова система Honeypot Proc), що дозволяє налаштовувати дерево / proc за допомогою UML, але це дуже довгий та складний процес схильний до людської помилки. Зловмисниками був розроблений інструмент під назвою UMLfp який допомагає визначити, чи перебуваєте ви в віртуальній машині UML.

1.3 Виявлення і атака на VMWare Honeypot

VMWare - це віртуальна машина подібна до UML, тому має дуже схожі проблеми щодо маскуванню віртуальної ідентичності апаратних засобів. У старих версіях VMware багато апаратних засобів не можна було налаштувати. Наступна таблиця показує, обладнання та налаштування за замовчуванням:

Hardware - Reported Type

Video Card - VMware Inc [VMware SVGA II] PCI Display Adapter

Network Interface Card - Advanced Micro Devices [AMD] 79c970

[PCnet 32 LANCE] (rev 10)

Hard disk - VMware Virtual IDE Hard Drive

CD Drive - NECVMWar VMware IDE CDR10

SCSI Controller - VMware SCSI Controller

Це, звичайно, робить виявлення пристосування VMWare надзвичайно простим. Щоб допомогти в боротьбі з цією проблемою, член французького проекту Honeyroot, Костя Корчинський, розробив патч, який дозволяє змінювати значення за замовчуванням. Наступним серйозним провалом приманок на базі VMWare є мережева MAC-адреса, яка прив'язана до мережевої карти.

Частина MAC-адреси (перші три октети) на інтерфейсі віртуальної мережі VMWare завжди одна з наступних трьох значень:

000569xxxxxx

000C29xxxxxx

005056xxxxxx

Це означає, що на машині Windows зловмисник може виконати команду "ipconfig / all" , а на машині UNIX "Ifconfig -a", щоб перевірити, чи є MAC-адреса віртуальним пристроєм. Виконання "arp -a" також дозволить зловмисникові у скануванні мережі для цих MAC-адрес.

Останньою слабкістю приманок на базі VMWare є бекдор вводу/виводу, який розробники залишили відкритими для конфігурація віртуальної машини під час виконання.

Нижче приведені декілька рядків коду збірки, за допомогою яким можна отримати доступ до цього бекдора:

```
mov eax, VMWARE_MAGIC ; 0x564D5868
```

```
mov ebx, b ; <Набір параметрів>
```

```
mov ecx, c ; <Набір параметрів>
```

```
mov edx, VMWARE_PORT ; 0x5658
```

```
in eax, dx
```

Деякі приклади команд, які можна використовувати через це:

04h - Отримати поточну позицію курсору миші.

05h - Встановити поточну позицію курсору миші.

06h - Отримати довжину даних у буфері обміну.

07h - Прочитати дані з буфера обміну.

08h - Встановити довжину даних для передачі в буфер обміну.

09h - Відправити дані на буфер обміну.

0Ah - Отримати версію VMware.

0Bh - Отримати інформацію про пристрій.

Виконавши якусь з представлених команд та перевіривши що вона була виконана успішно, зломисник може констатувати що він знаходяться у

віртуальній машині VMWare. Використання цього бекдора також може допомогти порушити роботу запущеної віртуальної машини VMWare. Цього можна досягти, використовуючи вразливість переповнення буфера. Ще один патч від Кості Корчинського змінює значення VMWARE_MAGIC у спробі приховати бекдор від злоумисника, хоча поки що не встановлено наскільки ефективним є цей метод.

1.4 Виявлення середовища chroot

Одним із поширених способів забезпечення бінарних пакунків і систем honeypot є використання середовища chroot jail. Найпростіший спосіб визначити, чи перебуваєте ви у середовищі chroot - це виконати команду «ls -lia» з кореневого каталогу. Слід взяти до уваги що кореневим каталогом є «.» та «..», відповідно до стандарту:

```
2 drwxrwxrwx  21 root   root    2096 Oct 16 18:17 .
```

```
2 drwxrwxrwx  21 root   root    2096 Oct 16 18:17 ..
```

Якщо ви перебуваєте в chroot і виконаєте ту саму команду результат буде іншим:

```
1553552 drwxrwxrwx   6 1000   100    4096 Dec 14 12:58 .
```

```
1553552 drwxrwxrwx   6 1000   100    4096 Dec 14 12:58 ..
```

1.5 Виявлення honeyd honeypot

Honeyd має різні функціональні можливості і дещо інші принципи роботи, тим не менш він має недоліки. Першою та головною проблемою honeyd є тенденція реагувати на пакети, на які він не повинен був відповідати. Якщо хтось посилає пошкоджений мережевий пакет на хост honeyd (наприклад пакет, який відкриває і відразу закриває з'єднання), хост honeyd все одно відповідатиме так якби він отримав дійсний пакет. Як правило, це буде пропущено інструментом

атаки, який не очікував почути відповідь. Щоб встановити цей факт, зловмисник повинен буде спостерігати за мережею за допомогою сніфера пакетів.

Наступним недоліком honeyd є те, як він обробляє відбитки операційної системи через стек TCP/IP. На жаль, аналіз мережі за допомогою інструменту nmap для знаходження мережевих пристроїв та визначення їх операційних систем не ідеальний і може давати помилкові позитивні результати (Valli, 2003). Якщо це станеться під час сканування зловмисником, це може викликати підозру.

Оскільки honeyd багато залежить від конфігурації заданої користувачем, необхідно враховувати концепцію людської помилки. Існує багато семантичних помилок, які можна зробити у файлі конфігурації honeyd. Ці помилки не можна виявити під час запуску оскільки технічно вони правильні. Помилка такого характеру, яка може бути зроблена у спробі емуляції серверу на базі OS Linux або Windows Server.

Остаточна проблема з honeyd є фактична емуляція сервісів. Оскільки це прості скрипти perl, зловмисник знову може отримати щось семантично або тематично неправильне і розпізнати приманку. Крім того надійність сценаріїв може відігравати певну роль у можливості розпізнавання honeyd.

Наприклад, скрипт routetelnet.pl який постачається за замовчуванням має тенденцію до збоїв з обробкою вхідних даних. Це може викликати підозру у зловмисника та підштовхнути продовжити розслідування.

1.6 Загальні проблеми з honeypot

Існує кілька факторів, які можуть створювати проблеми для honeypot в цілому. Більшість з них зосереджені навколо часу. Honeypot погано масштабуються під час атак з великою кількістю атакуючих хостів і достатня

пропускна здатність. Отже, вони страждають значним зниження продуктивності під час тривалої атаки. Також це може призвести до відмови в обслуговуванні.

Крім того, емульовані команди повинні добре імітувати вихідні команди, щоб не викликати підозру. Це непросте завдання, коли ви маєте справу зі зловмисником, який має глибокі, експертні знання про сервіс, протокол або операційну систему.

Все, що кодується людьми, може бути поставлено під загрозу. Протягом багатьох років це стосується різних програм, таких як брандмауери, веб-сервери та інші. Щоразу, коли створена нова програма, через певний проміжок часу можна очікувати звіти про помилки та вразливості. Honeypot не є виключенням. Ми повинні розуміти, що у кожному випущеному Honeypot, є відомі (і невідомі) вразливості. Як і в будь-якій іншій технології, необхідно вжити заходів для захисту від невідомих атак. У приманках з низькою взаємодією ризик є дещо обмеженим, оскільки зловмисники взаємодіють лише з емульованими сервісами, тобто, їм не надаються реальні програми та операційні системи для експлуатації. Однак, ми повинні припустити, що зловмисник може обійти контрольоване середовище емульованих служб, і саме тому необхідно зробити все для захисту програми від приманки. Для приманок з низькою взаємодією Win32 (наприклад, KFSensor) ви маєте встановити захищену базову ОС з останніми оновленнями, відключивши всі інші сервіси. Можливо, навіть встановити брандмауер на базі хоста: який дозволяє вхідні з'єднання з будь-яким портом, який здійснює моніторинг, але блокує всі інші вхідні з'єднання. Ще більш важливим є -

встановити блок брандмауера про будь-які ініційовані з'єднання. Це допоможе захиститись від загрози приманки при компрометації. Для приманок з низькою взаємодією Unix ми можемо вжити більших заходів. Середовище Chroot - це один звичайний спосіб поліпшити стримування проти атакованих процесів у системі Unix. Середовище Jail пропонує реальний спосіб обмежити те, що можна побачити в процесах. Патчі ядра низького рівня, такі як Systrace (дискреційний контроль доступу на основі процесу) або Grsecurity (Обов'язковий контроль доступу, заснований на процесах, захист адресного простору тощо) або інші, такі як SE Linux, які повинні використовуватися у приманках із низькою взаємодією, щоб захиститися від відомих і невідомих атак.

Для приманок з високою взаємодією ця проблема постає є більш складною. Тому що рішення вже дають взаємодію з реальною операційною системою та програма, внаслідок чого зловмисники мають більший простір для творчості і загальний ризик зростає. Якщо зловмисники можуть зламати приманку. Це означає, що потрібно ввести зовнішні заходи контролю даних, такі як IPS (система запобігання вторгнень) або обмеження пропускну здатності. У цих випадках можна зробити два кроки. Спочатку використовуйте декілька шарів управління. Це запобігає ризику виникнення єдиної точки відмови. Друге - можливість втручання людини. Приманки з високою взаємодією слід ретельно контролювати. Щоразу, коли на вашому honeypot відбувається аномальна активність (вихідні з'єднання, завантажені файли, посилена активність у системі,

нові процеси, системні входу тощо) необхідно сповіщувати адміністратора безпеки. Людина повинна контролювати все, що відбувається в системі. Щоразу, коли дія зловмисника перевищує поріг ризику вашої організації, ви можете заблокувати з'єднання між honeypot і зловмисником: скинувши пакети, перенаправити з'єднання тощо. Перевага адміністратора безпеки, який буде слідкувати за процесом полягає в тому, що ви вона може виявити діяльність, яку автоматичні механізми можуть пропустити. Це також дає надає більший контроль над тим, що робить зловмисник.

Традиційно більшість розгортань приманок не орієнтовані на конкретну ціль, натомість вони є загальними системами, розгорнутими у зовнішніх мережах. Це схоже на лов риби на будь-якому місці великого океану. Розгорнуті приманки мають потенціал для захоплення великої кількості риби. Організації зазвичай не сильно турбують автоматизовані або загальновідомі напади, їх більше лякають нові, невідомі досі вразливості, які ставлять під загрозу безпеку інформації. Honeypot мають бути розташовані в потрібному місці, в потрібний час та з правильною приманкою. Для цього такі приманки потрібно налаштовувати належним чином, що є набагато складнішою роботою.

1.7 Висновки до розділу 1

Honeypot зарекомендував себе як цінний інструмент для дослідження нових векторів атак і шкідливих програм. Він добре допомагає у ідентифікації та визначенні типів атак та вразливостей. На жаль, зловмисники знають про існування honeypot і розвивають контрзаходи для їх виявлення та зламу.

2 ВІДОМОСТІ ПРО МЕТОДИ РОЗВ'ЯЗАННЯ

2.1 Порівняльний аналіз Honeypot з конкурентними рішеннями

Honeypot проти Sandboxes. Sandboxes або Пісочниці - це інструменти, які використовуються для автоматизованого аналізу поведінки потенційних шкідливих програм в ізолюваному фізичному або частіше віртуальному середовищі. Типова пісочниця відкриває файл для аналізу або документ із відповідним зчитувачем та відстежує всі зміни та взаємодії, спричинені в системі. Зокрема, вона надає інформацію про зміни у файловій системі, реєстрі, процесах, завантажених бібліотеках, а також захопленому мережевому трафіку. Пісочниці різняться за типом проведеного аналізу, рівнем контролю деталей тощо. Вони обладнані інструментально, що дозволяє контролювати надану інформацію.

Зазвичай пісочниці використовуються для аналізу бінарних файлів. Але все частіше ці інструменти використовуються також для аналізу документів та веб-сторінок. Це означає, що методи пісочниці можна використовувати для виявлення та аналізу загроз, орієнтованих на клієнтські програми. Отже, деякі пісочниці можуть пропонувати функціональність, схожу за до роботи Honeypot на стороні клієнта.

Основна відмінність між пісочницею та приманкою полягає у цілях використання. Пісочниці більш орієнтовані на глибокий аналіз процесу зараження та дій, які виконуються зловмисними програмами після цього, тоді як мета

приманки - визначити, чи є щось шкідливим, в першу чергу, а вже потім - визначити механізми, що ведуть до зараження. Приманки стежать за тим, що відбувається після успішної інфекції але не фокусуються над цим.

Також можна відзначити, що пісочниці відрізняються від класичних приманок з високою взаємодією з точки зору їх складової ізоляції. Якщо частина зловмисного програмного забезпечення намагається зв'язатися з сервером IRC з пісочної скриньки, нічого, швидше за все, не станеться, якщо ви не створите у пісочниці необхідний об'єкт, який зможе забезпечити взаємодію, яку шукає зловмисне програмне забезпечення. Очевидно, що приманки з високою взаємодією буде спілкуватися безпосередньо з реальним сервером IRC.

На практиці можливе ще одне розмежування двох понять на основі їх режиму роботи. Зазвичай пісочниці працюють трохи довше, ніж приманки. Тому, що їх мета полягає у зосередженні уваги на аналізі поведінки після того, як відбулася інфекція.

Підсумовуючи, пісочниці та медоносні засоби є досить подібними інструментами, але вони відрізняються за своїм призначенням. Насправді до них слід ставитися як до двох допоміжних інструментів, які здатні співпрацювати. Приманки зосереджуються на механізмах, що призводять до зараження, тоді як пісочниці виконують глибокий аналіз шкідливих програм та дії, які відбуваються після зараження. Приклад послідовності подій, що демонструють таку співпрацю, може бути таким: спочатку приманка клієнт аналізує веб-сайт, а після отримання

підозрілого файлу відправляє його в пісочницю для подальшого тривалого аналізу (наприклад, відстеження ботнету).

Honeypots проти network telescopes. Network telescopes використовуються для спостереження та вивчення масштабних подій, наприклад моделей розповсюдження хробаків, а не конкретних атак чи вразливих місць, якими може скористатися будь-яка шкідлива програма. Як і приманки, невикористаний простір IP організації може бути використаний для створення цього рішення. Весь трафік, спрямований на network telescopes за походженням є підозрілим. Однак, на відміну від приманок, ці мережі не беруть участі у будь-якій формі взаємодії із вхідним трафіком. Тобто, вони пасивні. Ще одна відмінність полягає в масштабі: щоб спостерігати за масштабними подіями, network telescopes зазвичай охоплюють набагато більші простори мережі ніж приманки. У деяких випадках, як і UCSD Network Telescope¹⁰, майже одна восьма частина адресного простору IPv4 компанії виділяється під network telescopes. Трафік, перенаправлений на темну мережу, зазвичай, включає автоматичні сканування, активацію черв'яків та скануючих ботів, а також зворотний розбір DDoS-атак.

Honeypots проти Intrusion Detection/Prevention systems. Система виявлення вторгнень (IDS) - це програмний компонент (часто інтегрований із апаратним пристроєм, особливо у випадку комерційних рішень), який контролює та аналізує мережевий трафік чи поведінку операційної системи на предмет

несанкціонованих чи зловмисних дій. Система IDS, як правило, працює в пасивному режимі: вона виявляє загрозу, реєструє інформацію та запускає попередження. Система запобігання вторгнень (IPS) схожа на IDS, але зазвичай працює в активному режимі: вона здатна блокувати шкідливу поведінку.

Honeypot також досить часто використовуються для виявлення вторгнень. Однак, він не може розглядатися як заміна продукту IDS. Honeypot - це ресурси, до яких, має змогу отримати доступ лише зловмисник, а не системи для моніторингу трафіку. Це спрощує проблему виявлення вторгнень: приманки за своєю суттю менш схильні до помилкових позитивних спрацювань, але, як правило, вони більш конкретні і, ймовірно, потребують більшої адміністративної витрати. З іншого боку, вони не виявлять жодної атаки, яка спрямована на виробничі ресурси (тобто не спрямована на приманку). Отже, вони також не зможуть заблокувати атаку, спрямовану на виробничий ресурс. Це означає, що системи IDS / IPS, таким чином, пропонують кращий, можливий захист від атак та типів атаки на мережу (ціною більш високої кількості помилкових спрацювань). Тому приманки та IDS / IPS можна розглядати, як доповнюючі технології: медові горщики можуть виявити атаки, які пропускаються IDS / IPS. З іншого боку, IDS / IPS можна використовувати, як частину системи для переадресації зловмисників, віддалених від виробничих ресурсів, до приманки.

Honeypot проти web security proxies. Веб-проксі-сервер має можливість перехоплювати та аналізувати весь HTTP-трафік між браузером та веб-сервером. З точки зору браузера, це може бути абсолютно прозорий процес. Проксі-сервери можна використовувати, як частину встановлення приманки, щоб краще розрізняти трафік, який надходить до та від зломисників. Наприклад, їх можна використовувати для впровадження чорних списків або правил виявлення вторгнень. Можемо зробити висновок, що web security proxies є лише можливим доповненням honeypot.

2.2 Архітектура розгорнутих приманок

Для дослідження було обрано HoneySMB, HoneyWEB-SQL, Cowrie. Під час проектування архітектури розгорнутого Honeypot було взято до уваги що система Honeypot повинна бути максимально реалістичною та повинна бути чітко відділена від діючих систем.

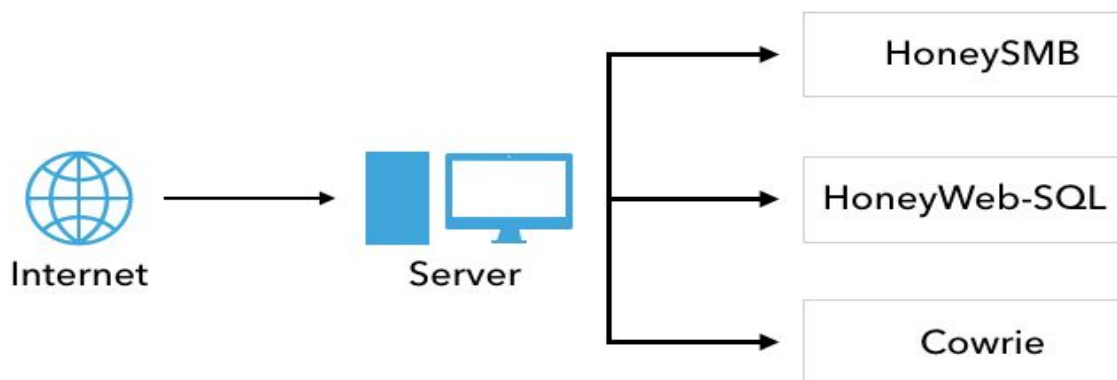


Рисунок 2.1 Схема розгорнутих Honeypot

Система була розгорнута на двох різних IP-адресах з використанням технологічних рішень Docker & ELK. На рис. 2.1 представлений загальний вигляд впровадженої системи з трьома різними приманками кожна з яких знаходиться в окремому контейнері. Стек ELK використовується для аналізу поточних журналів. Як показано на рис. 2.1, сервер використовується для зв'язку з клієнтськими програмами: HoneyWEB-SQL, HoneySMB і Cowrie. Вся система складається з брандмауэра, Linux сервера на базі Ubuntu, Elasticsearch, Logstash, Kibana та трьох відокремлених клієнтських програм, що працюють у віртуальному середовищі HoneySMB, HoneyWEB-SQL та Cowrie.

2.3 Розгортання HoneySMB

HoneySMB - це просте рішення яке імітує сервіси SMB. Дозволяє обмінюватися даними та реєструвати всі дії які відбулись. HoneySMB забезпечує запис без доступу до спільних накопичувачів користувачів із використанням трьох різних робочих груп: TMP, \$ IPC та DEMO. Також ці три робочі групи можуть бути налаштовані відповідно до необхідних цілей. Ці робочі групи містять зразок спільного накопичувача, на який завантажено декілька фіктивних файлів різних типів.

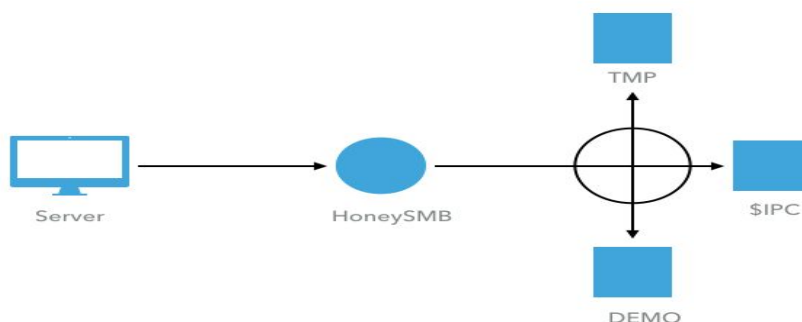


Рисунок 2.2 Архітектура HoneySMB

Усі сеанси будуть зберігатись у файлі ".pcap", який можна відкрити за допомогою програми Wireshark. HoneySMB навмисно налаштований таким чином, щоб була можливість для підключення будь-якого користувача з мережі Інтернет без необхідності введення пароля. Це рішення було спроектоване таким чином, щоб збільшити кількість можливих атак. Під час реєстрації всі ".pcap" файли будуть передані службовому виклику мережевого аналізатора Zeek. Zeek Network Security Monitor - потужний мережевий аналізатор. Для нього можна написати спеціальні дуже гнучкі за можливостями скрипти, які можуть витягнути необхідну інформацію з ".pcap" файлу та відобразити її у потрібній формі.

2.4 Розгортання HoneyWEB-SQL

HoneyWEB-SQL - це система honeypot для протоколу http. HoneyWEB-SQL виявляє вразливість SQL ін'єкції та дає змогу аналізувати вектори та методи які були використані під час атаки на базу даних SQL. Перш за все, він клонує веб-сайт на Docker-compose для його статичного розміщення, потім додає фіктивну сторінку входу, яка автоматично містить найпростішу вразливість "error based" SQL інфекції. Він також додає гіперпосилання можливість входу у файл index.html. У фоновому режимі фіктивна сторінка входу підключається до імітованої бази даних SQL, яка не є справжньою. База даних MySQL, яка використовується також має можливість реєструвати всі запити. Порт Docker-compose 80 відображається як 80 порт фактичної системи. Архітектура загальної системи наведена на рис. 2.3.

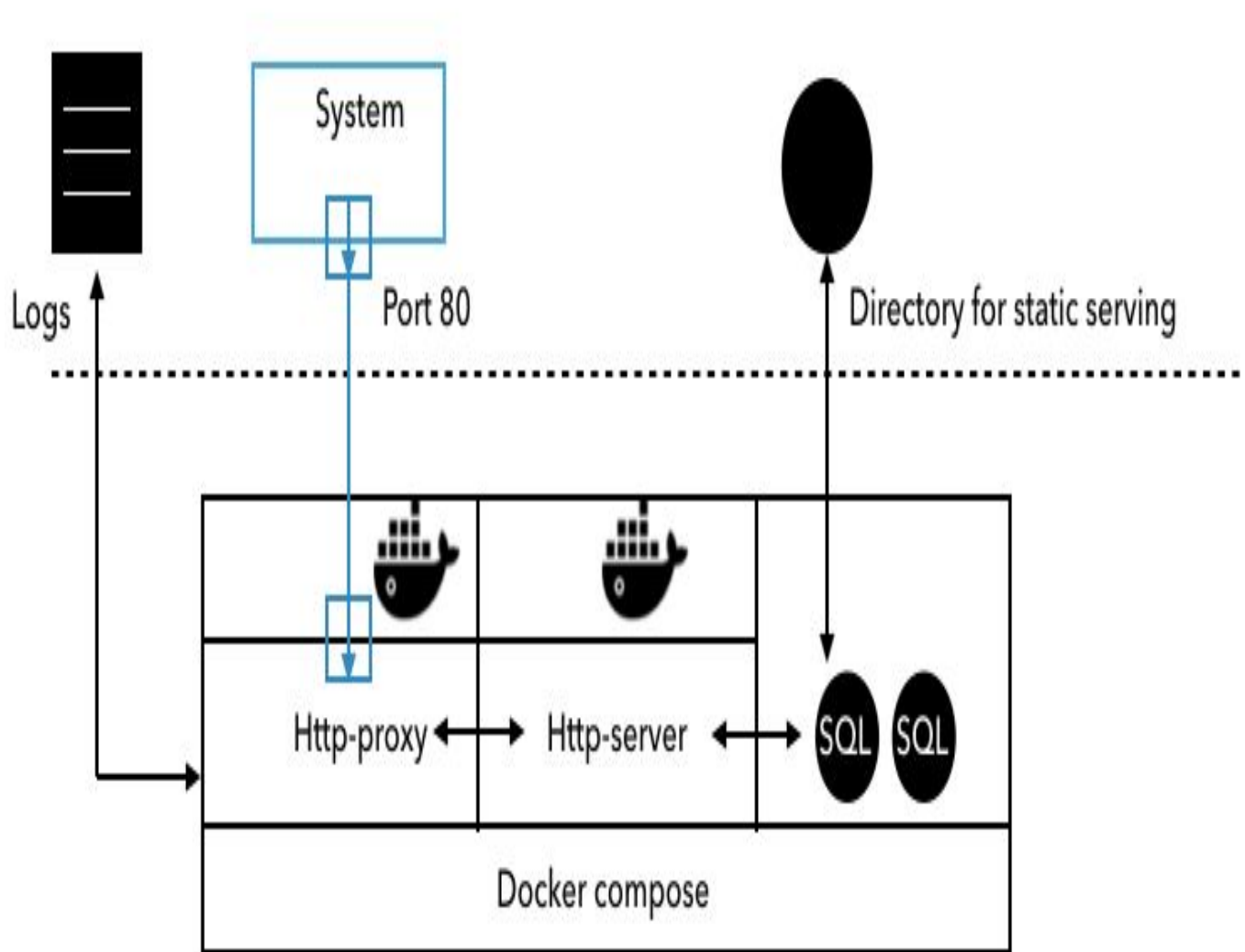


Рисунок 2.3 Архітектура HoneyWEB-SQL

Коли прийде запит на порт 80, будуть виконані наступні кроки http-сервером:

- Зловмисник намагається проєсплотувати існуючі вразливості на веб сервері відправляючи шкідливі запит.
- Якщо маємо помилку 404, http-сервер знову надсилає початкову сторінку index.html.

2.5 Розгортання Cowrie

Cowrie - це honeypot який емулює SSH-сервер, зі стандартними обліковими даними для входу наприклад: root / root, administrator / administrator, admin / admin та інші. Щойно зловмисник потрапить у систему, він отримає доступ до сфабрикованої оболонки Linux, яка має вигляд справжньої. В цій оболонці зловмисник може виконувати команди і отримати реалістичні відповіді подібні до відповідей, які він би отримав у звичайної користувацької системи. Honeypot Cowrie ізольований таким чином щоб зловмисник ніколи не зміг виконати команди не в емуляваному середовищі. Ізоляція досягається за рахунок того що Cowrie не побудований на базі, тої чи іншої, версії операційної системи взагалі. Середовище приманки Cowrie повністю побудоване та реалізоване за допомогою програмної мови Python.

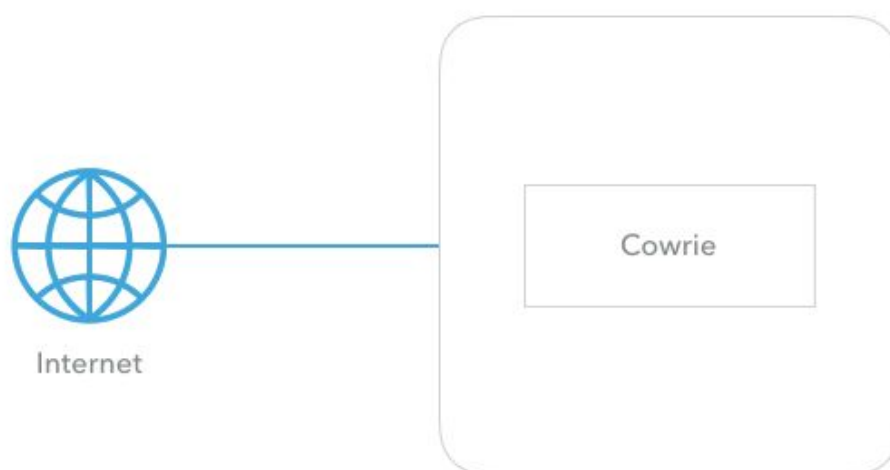


Рисунок 2.4 Архітектура Honeypot Cowrie

Cowrie побудований та є продовженням Honeypot Kirro, який мав значну популярність раніше. Cowrie був створений для того щоб виправити найбільші проблеми Honeypot Kirro та покращити якість роботи в цілому.

Подібно до інших приманок, Cowrie імітує роботу звичайного сервера, а також має можливість за допомогою вбудованих функцій аналізувати атаки які були здійснені на нього виходячи з існуючих логів. Це дозволяє адміністратору безпеки дуже швидко отримати представлення, які саме ймовірно атаки були здійснені, їх загальний успіх, частоту їх невдач, географічне розташування IP-адрес, яке у свою чергу, може дати представлення звідки саме проводилася атака. Також Cowrie може бути налаштований таким чином щоб збирати інформацію лише про одну конкретну або набір цілий IP-адрес.

2.6 Розгортання ELK Stack

ELK розшифровується як elasticsearch, logstash і kibana. Раніше це були три самостійні продукти, але в якийсь момент вони стали незалежними продуктами однієї компанії але продовжили розвиватись в одному напрямку. Кожен з цих інструментів є повноцінною, незалежною програмне забезпечення з відкритим програмним кодом, а всі разом вони складають потужне рішення для широкого спектра задач збору, зберігання і аналізу даних.

Logstash - це утиліта для збору, фільтрації і подальшого перенаправлення зібраної інформації до кінцевого сховища даних. Це рішення схоже на популярний програмний продукт fluentd. З тією різницею, що він написаний на ruby і трохи краще працює з elasticsearch, який є продуктом однієї і тої самої компанії.

Elasticsearch - це рішення для пошуку, побудованого поверх Apache Lucene, але з додатковими зручностями. Основними перевагами цього програмного продукту є легке масштабування, можливість реплікації. Ці переваги

виокремлюють його серед схожих аналогічних продуктів, вони зробили elasticsearch дуже зручним і гарним рішенням для високонавантажених проектів з великими обсягами даних.

Особливо доставляє в elasticsearch його простота і працездатність з коробки. Конфігурація за замовчуванням швидше за все буде працювати як треба для проектів середньої і відносно високої навантаженості. При цьому навколо ES склалося відмінне співтовариство, яке завжди підкаже, як правильно налаштувати ваш ES-кластер для вашої конкретної задачі.

Kibana - це user friendly інтерфейс, для Elasticsearch, який має велику кількість можливостей щодо пошуку даних в нетрах індексів Elasticsearch і відображенню цих даних в легкому для читання вигляді: таблицях, графіках і діаграмах.

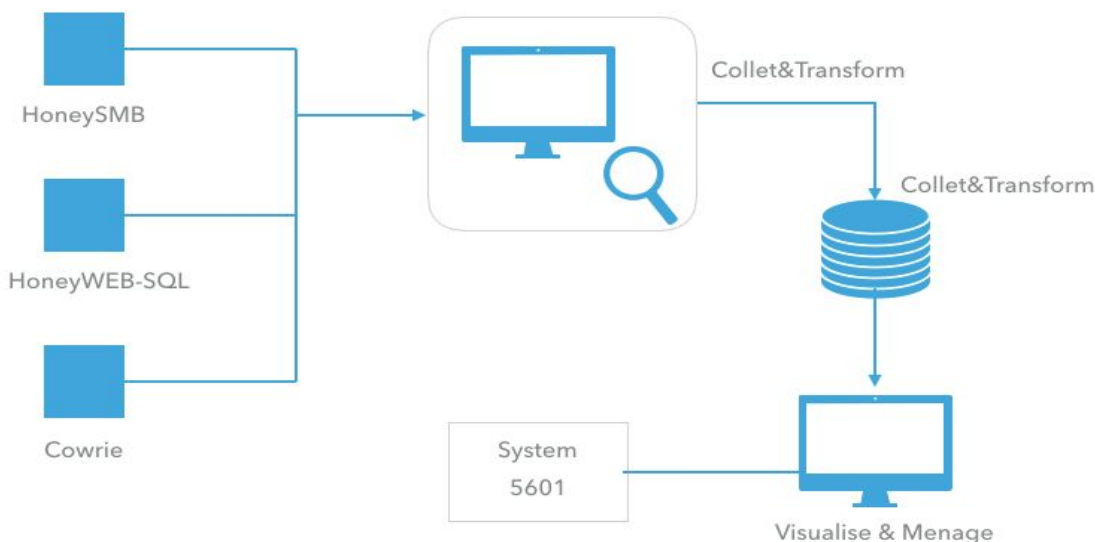


Рисунок 2.5 Архітектура ELK Stack

Дані журналу з усіх приманок захоплюються Logstash, далі ці неструктуровані дані перетворюються в структуровані та відправляються в Elasticsearch, він у свою чергу зберігає дані та індексує їх. Нарешті, Kibana використовується для візуалізації даних кінцевому користувачеві. Результат відображається за допомогою веб-інтерфейсу на порту 5601.

2.7.1 Аналіз результатів отриманих з HoneySMB

HoneySMB було розгорнуто в загальнодоступній мережі. Статистика цієї приманки містить кількість атак, унікальних IP-адрес, кількість завантажених шкідливих та кількість користувачів які намагалися встановити що вони знаходяться в приманці.

Таблиця 2.1, Статистика HoneySMB

Статистика	Кількість
Кількість атак	169
Кількість унікальних IP адрес	38
Кількість завантажених шкідливих файлів	8
Кількість користувачів які намагалися встановити що вони знаходяться в приманці	4
Кількість користувачів яким вдалося встановити що вони знаходяться в приманці	2

Протокол SMB використовується в основному в операційній системі Windows, але його можна використовувати і в операційній системі Linux, враховуючи це, зловмисники також додали декілька шкідливих бінарних файлів Linux у спільний SMB-накопичувач, щоб викликати атаку drive-by-download. На приманку HoneySMB було завантажено 8 різних шкідливих файлів, з яких 6 - для ОС Windows, а решта - для ОС Linux. Більшість зловмисників намагались завантажити бекдор, в той час, як інші намагались завантажити шкідливі програми за допомогою яких хотіли атакувати інші ресурси створюючи атаку відмову у обслуговуванні. Лише чотирьом з восьми зловмисників вдалося не лише загрузити бекдор а і скористатися ним, тобто зробити зворотнє з'єднання з сервером. Всього двоє зловмисників ймовірно змогли встановити, що система є приманкою тому що після, під час, аналізу логів було виявлені команди за допомогою яких можна було встановити віртуальну складову та після виконання цих команд більше не було ніякої активності з цих ір адрес.

Для більш детального розгляду необхідно ознайомитись з звичайною комунікацією SMB та можливих атак. У звичайному зв'язку NTLM, клієнт ініціює з'єднання, надсилаючи ім'я користувача на SMB-сервер. Натомість SMB-сервер запитує аутентифікацію, він надсилає виклик, в якому користувач-клієнт повинен зашифрувати його своїм хешем і знову відправити на сервер. Потім сервер намагається розшифрувати цей виклик, якщо він успішно розшифровує його, тоді клієнту буде наданий доступ, інакше сервер SMB відхилить автентифікацію клієнта.

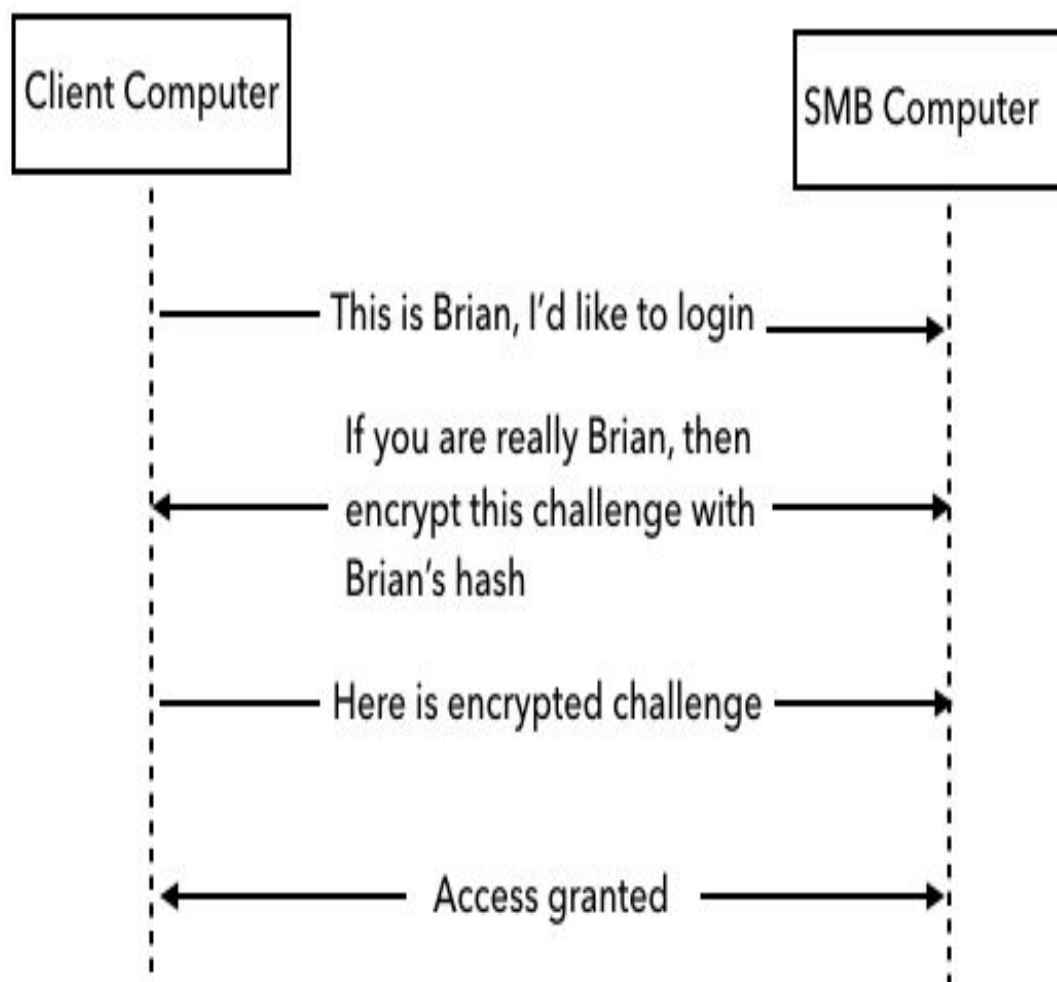


Рисунок 2.6 SMB комунікація

Під час атаки ретрансляції SMB зломисник розміщується між клієнтом і сервером, як у сценарії "людина посередині". Тепер, коли клієнти намагаються під'єднатись до SMB-сервера, пакет захоплюється зломисником, і він відповідає на нього як SMB-сервер. Зробивши тристороннього рукошлявання, зломисник відмовив у доступі клієнту, хоча він фактично використовує облікові дані клієнтів для входу на сервер SMB. Таким чином, аутентифікація не захищена від приведеної атаки на SMB. Ця вразливість існує в протоколі NTLM V2. Весь процес показаний на рис. 2.7 .

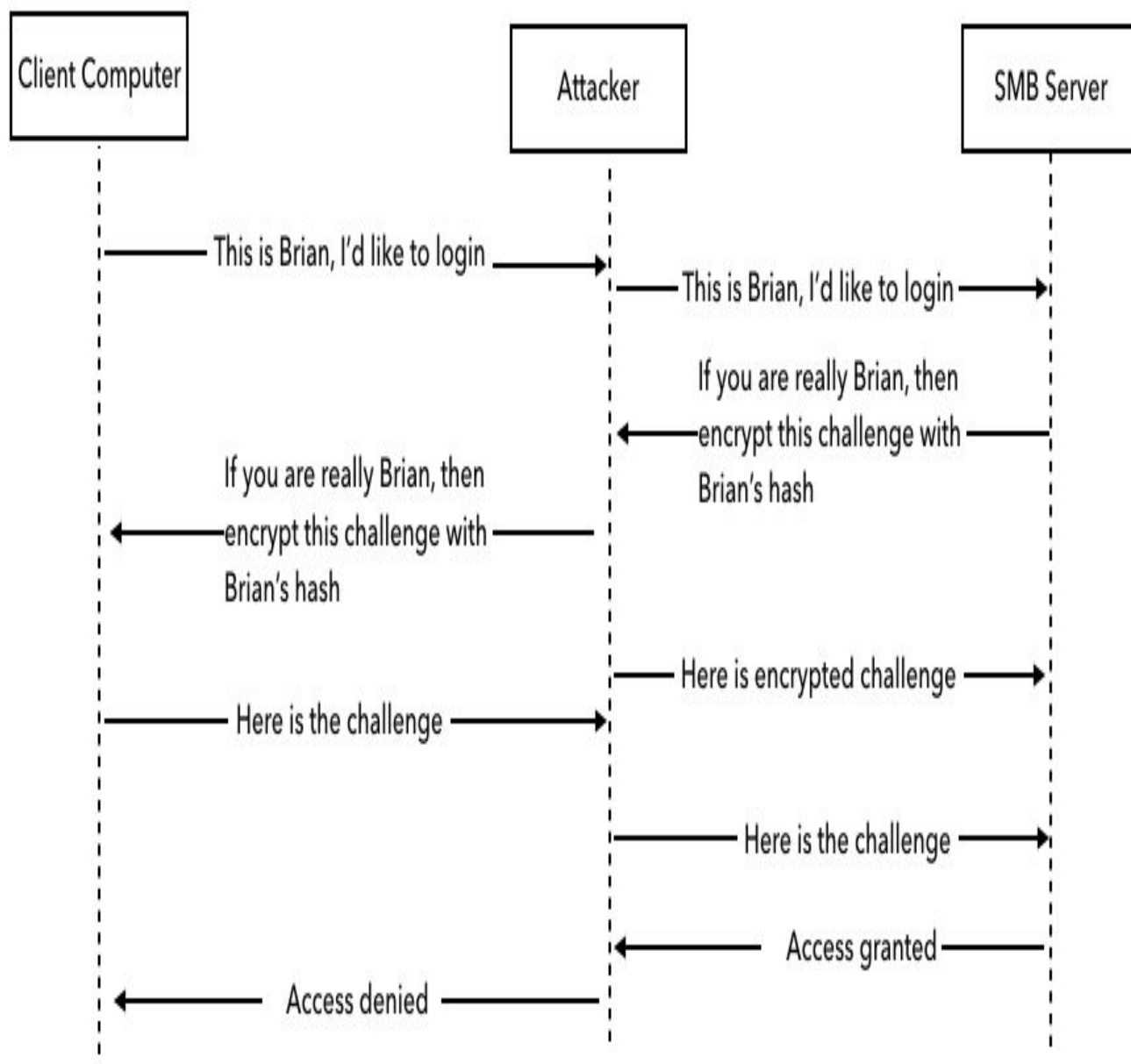


Рисунок 2.7 Сценарій атаки на SMB

Проаналізувавши нормальну роботу SMB та можливу атаку, можна краще зрозуміти дії зловмисників. Треба зазначити. Що представлена атака є загальновідомою для версії NTLM V2, тому можна було скористатися будь-яким відомим, безкоштовним сканером вразливостей. Єдиною вимогою до нього є останнє оновлення версії та бази даних вразливостей.

2.7.1 Аналіз результатів отриманих з HoneyWEB-SQL

Таблиця 2.2 Статистика HoneyWEB-SQL

Статистика	Кількість
Кількість атак	287
Кількість унікальних IP адрес	32
Кількість завантажених шкідливих файлів	2
Кількість користувачів які намагалися встановити що вони знаходяться в приманці	0
Кількість користувачів яким вдалося встановити що вони знаходяться в приманці	0

Атаки які були здійснені зловмисниками приведені нижче:

1) Shell Request

Дві IP-адреси надсилають наступний GET-запит, щоб перевірити, чи веб-сервер використовує функцію `php exec ()` чи ні.

`shell?%75%6E%61%6D%65%20%2D%61` якщо декодувати з ASCII то маємо `"uname -a"`

2) BEAST Attack

Атака BEAST - це коротка форма експлуатації браузера націлена проти SSL / TLS. Він націлений на експлуатування слабкої сторони CBC (Cipher Block Chain) для переходу користувача до наявного протоколу SSL. Ця вразливість у CBC може призвести до атаки "людина посередині" на протокол SSL.

3) php-myadmin

Декілька IP-адрес надсилають шкідливі запити на використання інтерфейсу `"phpmyadmin"`, доступного на веб-сервері Apache.

Ці запити націлені на те щоб отримати доступ до бази даних `"mysql"` за допомогою певних запитів.

4) Сканування портів

Більше 30 IP-адрес використовували сканери портів.

5) Сканування Domain

Більше 25 IP-адрес використовували сканери для знаходження sub-domain.

Жодному зі зловмисників не вдалося встановити, що він знаходиться у HoneyPot, основний час вони намагалися отримати доступ до бази даних, а після досліджували її.

2.7.3 Аналіз результатів отриманих з Cowrie

Таблиця 2.3, Статистика HoneyDB

Статистика	Кількість
Кількість атак	376
Кількість унікальних IP адрес	45
Кількість завантажених шкідливих файлів	0
Кількість користувачів які намагалися встановити що вони знаходяться в приманці	0
Кількість користувачів яким вдалося встановити що вони знаходяться в приманці	0

Ця приманка є найбільш вдалою серед усіх трьох за кількістю здійснених атак. За цей час було зареєстровано понад 376 атак та 45 унікальних IP-адрес.

Приманка була розгорнута за двома сценаріями:

- Облікові дані до першого сценарію: administrator / administrator
- Облікові дані до другого сценарію: root / root

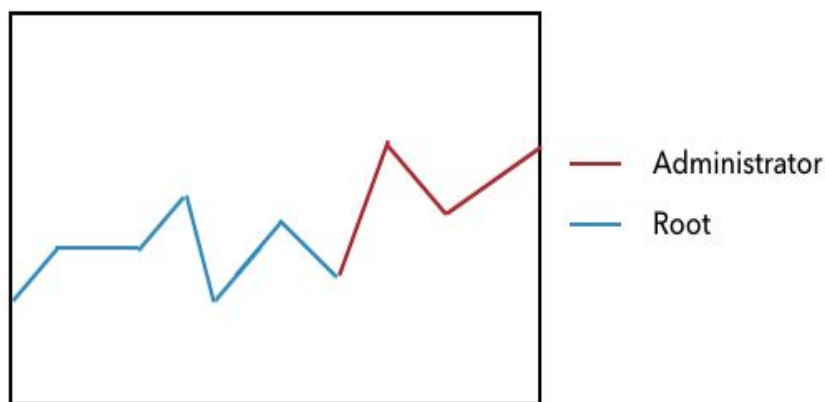


Рисунок 2.8 Кількість атак у день після розгортання

Приманку було розгорнуто на 3 дні із використанням даних root/root та на 4 дні з використанням administrator/administrator. Хоча administrator був привілейованим користувачем трохи більше часу, лише 109 запитів на з'єднання були з administrator у якості імені та паролю користувача, а всі інші запити надходили для root.

2.8 Висновки до розділу 2

Слабким місцем будь-якого Honeypot є його віртуальна складова проаналізувавши яку можна встановити що система є приманкою. Під час дослідження було встановлено що лише на одному з трьох розгорнутих приманок HoneySMB декілька користувачів ймовірно змогли встановити що вони знаходяться у Honeypot. Щодо двох приманки HoneyWEB-SQL, то зловмисники намагалися знайти і поексплуатувати вразливість та після цього отримати зміст бази даних, після цього вони в основному припинили дослідження системи. Cowrie був найбільше атакований під час дослідження також, що цікаво, він є найбільш поширеною приманкою.

3 МЕТОДИ ВИЯВЛЕННЯ HONEYPOT

3.1 Узагальнення методів виявлення Honeypot

Серед усіх найбільш поширених реалізацій Honeypot, які були розглянуті та проаналізовані у даній роботі можна прийти до висновку, що всі вони реалізовані для емуляції того чи іншого сервісу, процесу тощо. Найчастіше ці рішення розгорнуті на базі віртуального середовища або застарілого обладнання.

Тому, завдання з виявлення Honeypot, фактично зводиться до завдань з виявлення значень за замовчуванням віртуальної складової та застарілого обладнання, яке в сучасних умовах, дуже сумнівно, що може використовуватись для задоволення потреб високотехнологічних рішень.

Виходячи з цього можна припустити, що методи виявлення Honeypot можна автоматизувати але є ряд проблем: різні реалізації Honeypot можуть використовувати різні засоби віртуалізації або не використовувати їх взагалі. Використання застарілого обладнання теж не є панацеєю, тому що є приманки, які можуть бути розгорнуті на найновішому залізі.

Тому розробка універсального рішення для виявлення будь-якого Honeypot є досить довгим та складним завданням, а враховуючи те, що технології постійно розвиваються та удосконалюються - над складним. Тому у даному розділі було вирішено розглянути: розглянути на якому обладнанні зазвичай розгорнуті Honeypot, які не використовують віртуальне середовище; створити програму для

розпізнавання Honeypot Cowrie, який виявився найбільш “цікавим” для зловмисників під час дослідження; запропонувати критерії стійкості до розпізнавання, за якими можна, було б, швидко оцінювати складність до виявлення тієї, чи іншої реалізації Honeypot.

3.2 Виявлення honeypot на базі застарілого обладнання

Виявлення Honeypot, розгорнутого на базі застарілого обладнання є не менш важливим завданням, хоча і за статистикою частіше приманку розгортають на базі віртуального середовища. Крім застарілого обладнання необхідно звертати увагу на версії протоколів, операційної системи, програмного забезпечення тощо. яке використовується на потенцій

Для того, щоб зрозуміти, яке обладнання можна вважати застарілим було проведено невелике дослідження, яке мало на меті знайти та проаналізувати, на якому саме залізі було розгорнуто honeypot. Під час нього було розглянуто та проаналізовано 18 honeypot які розгорнуті в мережі інтернет. Всі розглянуті honeypot були знайдені за допомогою пошукової системи Shodan. Найбільше розгорнутих honeypot знаходяться у США, Німеччині, Франції, Польщі та Великобританії.



Рисунок 3.1 Кількість розгорнутих Honeypot

Таблиця 3.1, Honeypot на базі застарілого обладнання

Кількість honeypot	Сервіси	Операційна система	Процесор	Кількість оперативної пам'яті, гб	IP
5	FTP, HTTP	Windows XP	Intel Pentium	2	208.69.109.8 85.16.52.227 74.208.157.79 91.121.56.197 195.76.33.244
4	FTP, HTTPS, HTTP, DNS, Telnet	Windows IIS	AMD Phenom II	3	34.245.230.157 91.121.56.197 208.98.206.242 185.121.24.128
3	HTTP	Ubuntu	-	-	92.37.68.17 66.240.210.72 141.92.130.227
2	DNS, HTTP	Windows XP	Intel Celeron	1	213.215.147.21 69.55.224.31
2	FTP, HTTP	Ubuntu	Athlon II 64 x2	2	82.165.131.213 71.6.177.225
1	HTTPs	CentOS	Intel i3	4	103.21.130.2

Проаналізувавши отримані дані можемо констатувати, що застаріле залізо дійсно використовується для розгортання honeypot. Потрібно відзначити, що частіше зустрічається операційна система Windows та процесори Intel, а найпопулярнішими сервісами є FTP та HTTP.

3.2 Розробка програми для виявлення Cowrie honeypot

Honeypot Cowrie виявився найбільш атакованим серед інших під час проведеного дослідження але жоден із зловмисників потенційно не зміг встановити, що він знаходився у приманці. Саме тому було прийнято рішення розробити програму для виявлення Honeypot Cowrie.

Для створення програми виявлення був проаналізований Honeypot Cowrie, враховані його недоліки та виділені чотири потенційний методи виявлення.

Першим методом є перевірка відповідності версії SSH. Honeypot Cowrie емулює роботу сервісу SSH але за замовчуванням використовує недостовірну версію сервісу. За замовчуванням, якщо версія SSH є неправильною у відповіді міститься помилка “bad_version”. Виходячи з цього було реалізовано дві функції: conToSSH(host, port) та probIncorrectVers(sockfd). Перша функція conToSSH(host, port) приймає ір адресу хосту, порт; відкриває з'єднання, зберігає відповідь, за замовчуванням використовується ір адреса локального серверу та порт 22. Друга функція probIncorrectVers(sockfd), закриває з'єднання та шукає у відповіді помилку “bad_version”. Якщо помилка знайдена, можна констатувати, що використовується недостовірна версія SSH та потенційно користувач може

знаходиться у Honeypot. Звісно, цієї проблеми можна уникнути сконфігурувавши Cowrie належним чином та вказавши правильну версію SSH. Далі програма переходить до другого методу виявлення Honeypot Cowrie.

Другий можливий метод виявлення Honeypot Cowrie виникає через його неправильну конфігурацію. Найпростіше це можна перевірити шукаючи помилки “packet corrupt” та “protocol mismatch” у відповіді. Для цього була створена та використана функція probPackets(sockfd), яка приймає відкрите з'єднання у якості параметру, зберігає відповідь, закриває з'єднання та перевіряє наявність помилок “corrupt” та “mismatch” у відповіді. Недоліком цього методу є те, що такі помилки також можуть траплятись через неправильне налаштування OpenSSH, тому можуть траплятись помилкові спрацювання. Якщо помилки були знайдені, можна допустити, що користувач можливо знаходиться у Honeypot.

Третім можливим методом виявлення Honeypot Cowrie є можливе дублювання банеру у відповіді. Для цього нам необхідно проаналізувати відповідь на присутність таких самих помилок як у “packet corrupt” та “protocol mismatch” як і у попередньому методі. Якщо помилки будуть знайдені, то можна констатувати, що користувач потенційно може знаходитися у Honeypot.

Четвертим та останнім методом виявлення є порівняння банеру SSH серверу зі списком банерів за замовчуванням, які використовуються у Honeypot Cowrie. Якщо ми маємо спрацювання всіх чотирьох методів, тоді можемо констатувати з

досить високим рівнем вірогідності, що користувач знаходиться у Honeypot Cowrie.

Повний код програми знаходиться у Додатку А.

Для тестування програми Honeypot Cowrie був розгорнутий локально. Для підтвердження концепції був обраний незалежний Honey Cowrie, який має ір адресу 109.199.20.58 та використовує порт 22. Незалежний Honey Cowrie був знайдений за допомогою пошукового сервісу Shodan.

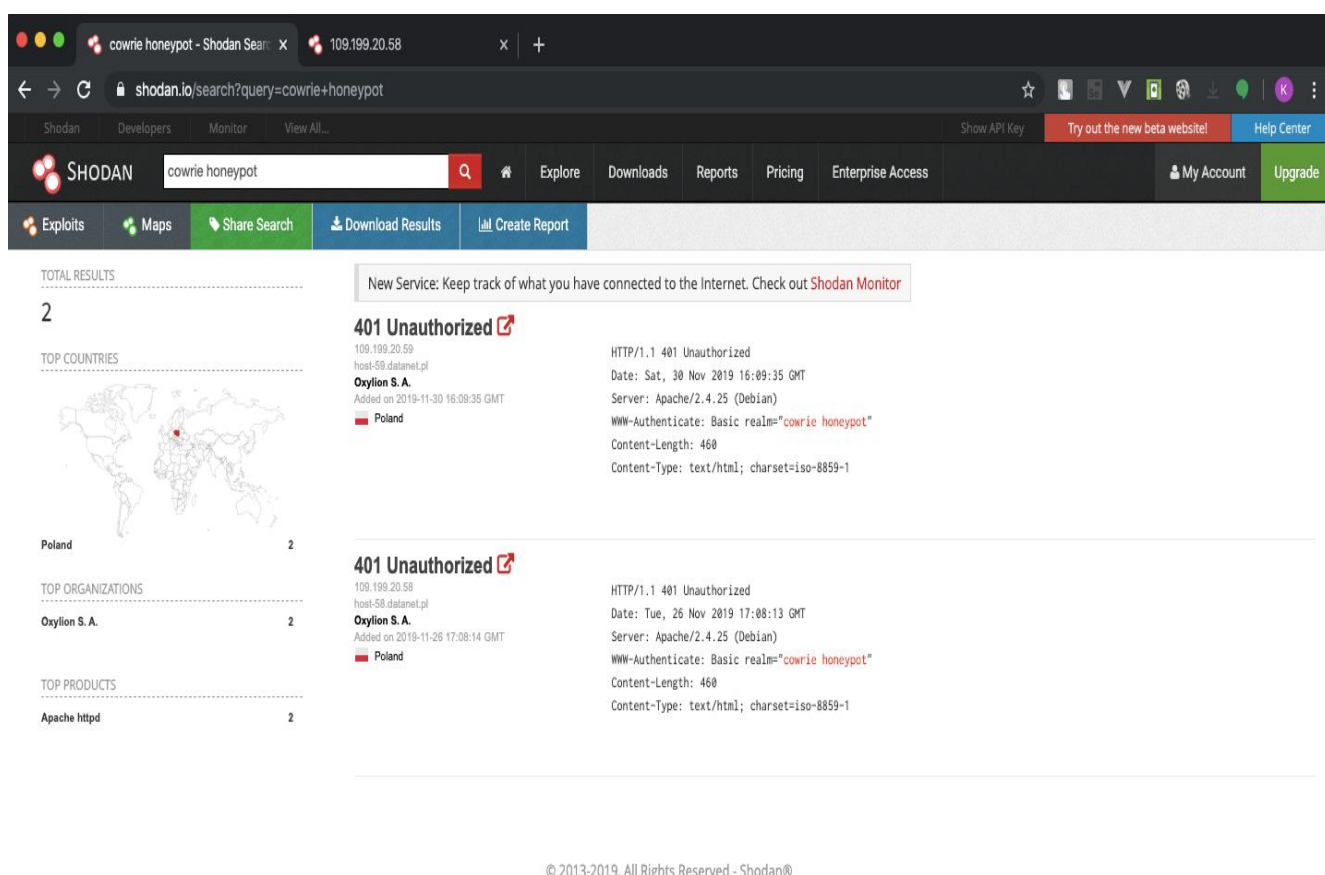
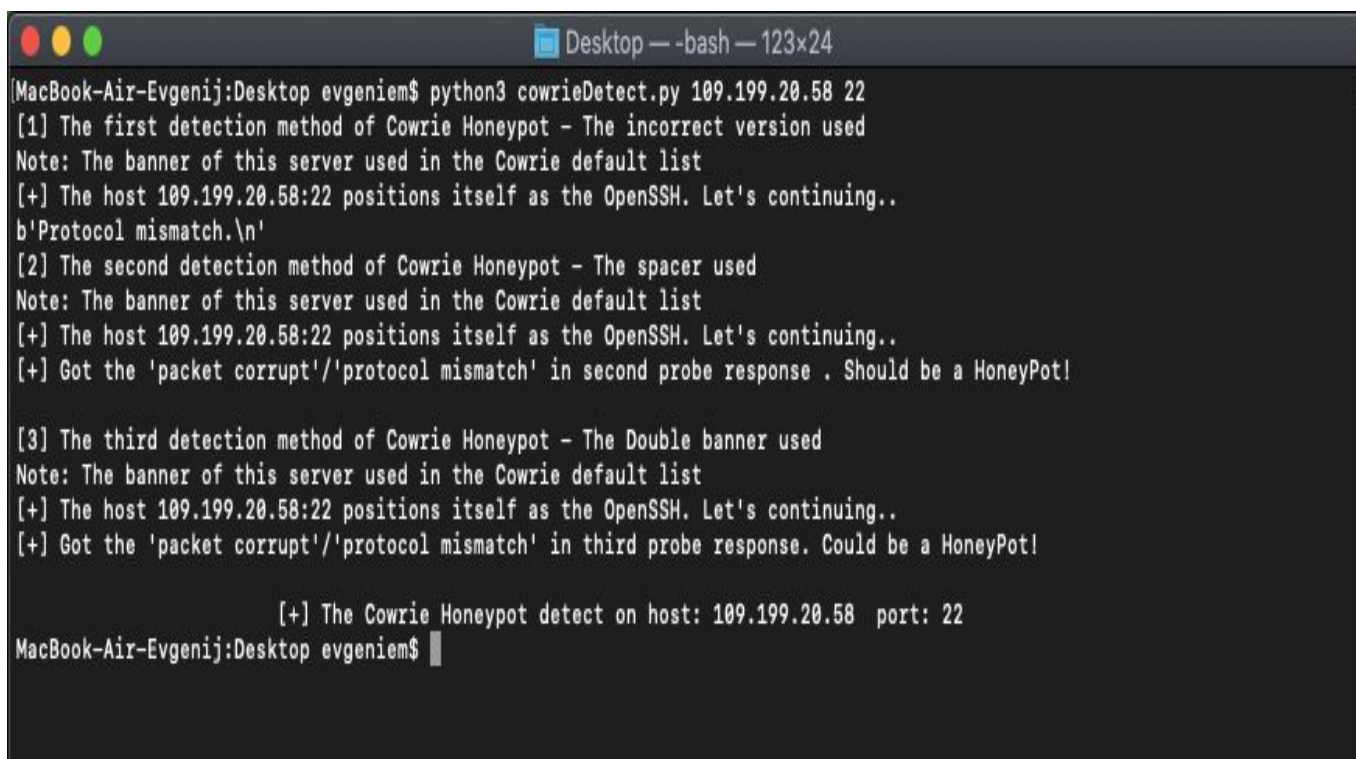


Рисунок 3.2 Пошук незалежного Honeypot Cowrie

Для автоматичної перевірки незалежного хосту на якому розгорнуто Honeypot Cowrie необхідно запустити створену програму за допомогою python3 та вказати два параметрами: ір адресу та port. Приклад виконання програми:

```
python3 cowrieDetect.py 109.199.20.58 22
```



```
MacBook-Air-Evgenij:Desktop evgeniem$ python3 cowrieDetect.py 109.199.20.58 22
[1] The first detection method of Cowrie Honeypot - The incorrect version used
Note: The banner of this server used in the Cowrie default list
[+] The host 109.199.20.58:22 positions itself as the OpenSSH. Let's continuing..
b'Protocol mismatch.\n'
[2] The second detection method of Cowrie Honeypot - The spacer used
Note: The banner of this server used in the Cowrie default list
[+] The host 109.199.20.58:22 positions itself as the OpenSSH. Let's continuing..
[+] Got the 'packet corrupt'/'protocol mismatch' in second probe response . Should be a HoneyPot!

[3] The third detection method of Cowrie Honeypot - The Double banner used
Note: The banner of this server used in the Cowrie default list
[+] The host 109.199.20.58:22 positions itself as the OpenSSH. Let's continuing..
[+] Got the 'packet corrupt'/'protocol mismatch' in third probe response. Could be a HoneyPot!

[+] The Cowrie Honeypot detect on host: 109.199.20.58 port: 22
MacBook-Air-Evgenij:Desktop evgeniem$
```

Рисунок 3.3 Результат роботи програми з виявлення Honeypot Cowrie

Проаналізувавши результат роботи програми ми бачимо, що всі три запропоновані методи виявлення Honeypot Cowrie спрацювали, тобто можна з високою вірогідністю сказати, що на даному хості розгорнутий Honeypot Cowrie. А через те, що був спеціально обраний Honeypot Cowrie можна констатувати, що програма відпрацювала правильно. Базуючись на отриманих результатах можемо констатувати, що методи розпізнавання є дієвими та правильно імплементовані у програмі.

3.3 Критерії стійкості honeypot до розпізнавання

Необхідність розробки критеріїв стійкості виникла через те, що більшість поточних реалізацій Honeypot не перевіряються, розробниками у процесі створення приманки на можливість виявлення. На мій погляд, причинами, через які Honeypot не перевіряється на можливість виявлення, під час розробки, є недостатнє розуміння, як саме це можна, спробувати, зробити та занижена важливість самого факту можливості виявлення. Тому було вирішено розробити критерії стійкості на основі даної роботи та загальних рішень(патчів, методів тощо) спрямованих на приховування того чи іншого типу Honeypot.

У таблиці 3.1 представлено чотири критерія стійкості до виявлення, які є дуже важливими при розробці та перевірці різних типів Honeypot на можливість виявлення.

Перший критерій(значення за замовчуванням) - є використання у Honeypot значень за замовчуванням, присутність яких, викликає підозру. Більш досвідчені злоумисники з легкістю можуть констатувати знаходження у Honeypot і припинити дослідження. Цей недолік є найпростішим, тим не менш, він досі є не виправним та присутній у більшості реалізацій Honeypot.

Другий критерій(наповнення системи) - полягає у відповіді на запитання “чи схожа приманка на звичайну систему, сервер тощо”. Зазвичай на більшості систем, які виконують певні функції, в залежності від застосування, присутність у достатній кількості директорій та файлів які, як правило, мають різний час

створення є досить вагомим. Також потрібно звернути увагу на встановлені програми їх версії та час останнього оновлення. Цей фактор є одним із ключовим при дослідженні системи особливо, якщо приманка була розгорнута на звичайному залізі або виявити присутність віртуальної складової не вдалося.

Третій критерій(наявність автоматизованих методів виявлення) - полягає у наявності будь-яких автоматизованих рішень, які спрямовані на можливість виявлення певного типу Honeypot. Створення автоматизованих рішень говорить про те, що було проведене дослідження, яке підтвердило можливість виявлення певного типу Honeypot за розробленим методом. Тобто можна допустити, що було привернути увагу розробників Honeypot того типу, для якого було створене це автоматизоване рішення виявлення.

Четвертий критерій(доповнення спрямовані нівелювати критерії 1 та 2) - полягає у наявності доповнень(патчів), які були розроблені для заміни значень за замовчуванням або для реалістичного наповнення системи. Цей критерій є не лише показником привернути уваги до проблем певного типу Honeypot, але й активності членів організації та ентузіастів. Також можливість швидко та легко підвищити стійкість до виявлення приманки є прямою перевагою над конкурентами.

У таблиці 3.3.1 наведені розроблені критерії стійкості до виявлення Honeypot. Якщо певний критерій можна виділити як пряму проблему відповідно до наведеного типу honeypot буде стояти позначка “+”, якщо ні “-”.

Таблиця 3.1, Критерії стійкості до виявлення Honeypot

Тип приманки	Перший критерій (значення за замовчуванням)	Другий критерій (наповнення системи)	Третій критерій (наявність автоматизованих методів виявлення)	Четвертий критерій (доповнення спрямовані нівелювати критерії 1 та 2)
UML	+	+	+	-
VMWare	+	+	-	+
Chroot	+	+	-	-
Honeyd honeypot	+	+	-	+
Honeypot розгорнутий на базі заліза	-	+	-	+

3.4 Висновки до розділу 3

У даному розділі було розглянуті загальні принципи на яких базуються методи виявлення Honeypot та приведена інформація на чому вони базуються та чим відрізняються. Було проведене невелике дослідження на предмет наявності серед загально доступних приманок таких, які розгорнуті на базі звичайного обладнання без присутності віртуальної складової та можливість їх виявлення за певними ознаками. Було встановлено, що частіше, дійсно використовується застаріле обладнання для розгортання приманок без використання віртуальної складової і виявлення за ознаками старого обладнання, операційної системи можливе. Також, була розроблена програма, яка автоматизовано може визначити чи є приманка Honeypot Cowrie та проведене незалежне випробування дієвості програми на незалежно розгорнутому Honeypot Cowrie знайденому за допомогою пошукової системи Shodan.

На базі даної роботи та загальновідомих методів виявлення Honeypot були розроблені критерії стійкості до виявлення. За створеними критеріями стійкості були проаналізовані основні типи реалізації honeypot. Результати аналізу запропоновані у вигляді таблиці.

4 РОЗРОБКА СТАРТАП-ПРОЕКТУ “АВТОМАТИЗАЦІЯ ВІЯВЛЕННЯ HONEYPOT”

4.1 Опис ідеї проекту

У даному розділі буде розглянуто економічне обґрунтування реалізації стартап-проекту “Автоматизація виявлення HONEYPOT”. Технологія буде реалізована у вигляді програми та надаватися користувачам за підпискою.

Розділ включає:

- впровадження технології;
- створення стратегії завоювання ринку та подальший розвиток проекту.

Таблиця 4.1, Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигода для користувачів
Ідея полягає у продажу підписки на программний продукт який буде розпізнавати найбільш часто використовувані Honeypot	Тестування розробниками Honeypot на предмет виявлення	Розробники отримують можливість тестувати Honeypot в процесі розробки, що дає їм змогу покращити стійкість Honeypot до виявлення
	Тестування спеціалістами з кібербезпеки та ентузіастами вже існуючих Honeypot	Для спеціалістів з кібербезпеки це можливість зекономити час, коли необхідно дослідити/вибрати той чи інший Honeypot. Для ентузіастів найчастіше основна проблема - недостатня кваліфікація

У таблиці 4.1 було описано головну ідею, можливі напрямки використання та цінність яку несе в собі кінцевий продукт. Реалізація продукту буде базуватися на критеріях стійкості до виявлення приведених в даній роботі, а також включати опції виявлення найбільш розповсюджених honeypot які базуються на віртуальному середовищі та застарілому обладнанні. Розробники та спеціалісти з кибер безпеки зможуть сканувати та тестувати розгорнутий honeypot на можливість виявлення.

Таблиця 4.2, Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	Товари/Концепції конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Конкурент1	Конкурент2	Конкурент3			
1	Форма реалізації	програма	програма	програма	програма			+
2	Собівартість	Низька	Низька	Середня	Висока			+
3	Виявлення найпоширеніших Honeypot з віртуальною складовою	Так	Так	Так	Так		+	
4	Виявлення Honeypot без віртуальної складової	Так	Ні	Ні	Ні			+
5	Крос-платформеність	Так	Так	Ні	Так	+		

У таблиці 4.2 наведені слабкі, нейтральні та сильні сторони потенційного продукту, що є основою його конкурентноздатності. Сильними сторонами є форма реалізації у вигляді програми, низька собівартість та можливість виявлення Honeypot без віртуальної складової. До слабких сторін можна віднести складність у реалізації та недостатня гнучкість у можливості виявлення нових або не поширених реалізацій Honeypot.

4.2 Технологічний аудит ідеї проекту

У даному підрозділі будуть описані технології які будуть використані під час реалізації проекту.

Таблиця 4.3, Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології реалізації	Наявність технології	Доступність технології
1	Створення програми	Python	Наявна	Безкоштовна
		Ruby	Наявна	Безкоштовна
		C++	Наявна	Безкоштовна
Для створення програми була обрана мова програмування Python				

Програма буде створена за допомогою мови програмування Python тому що має необхідні бібліотеки для значного спрощення реалізації і є простішою у використанні.

4.3 Аналіз ринкових можливостей стартап-проекту

Аналіз ринкових можливостей дає змогу зрозуміти у якому стані знаходиться ринок та його динаміка, наявність перешкод та головних конкурентів. Також за його допомогою можна змодельовати потенційні напрямки розвитку стартапу.

Таблиця 4.4, Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн/ум.од	50000
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі, %	$R = (3000000 * 100) / (1000000 * 12) = 25\%$

Рентабельність галузі за середніми показниками більша ніж можливий прибуток від вкладання грошей на депозит в банку, тому стартап є привабливим для потенційного інвестора. Також потрібно зазначити, що немає ніяких

обмежень для виходу стартапу на ринок та перешкод у вигляді необхідність проходження специфічної сертифікації.

У таблиці нижче були визначені можливі групи потенційних клієнтів та перелік вимог.

Таблиця 4.5, Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія	Відмінність у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Програма для виявлення найпоширеніших Honeypot	Компанії, розробники, фахівці з кібербезпеки	Цільова група намагається розробити/визначити найбільш стійкий до розпізнавання Honeypot	Програма повинна бути легкою у використанні, швидкою та надійною

Була відзначена характеристика потенційних клієнтів стартапу та основну потребу яку формує ринок - зручна, швидка програма для виявлення honeypot. Основними сегментами ринку можуть бути: великі, середні та малі компанії які використовують або збираються використовувати honeypot, а також розробники honeypot, фахівці з кібербезпеки та ентузіасти.

Нижче сформовані таблиці які містять фактори, що допомагають у ринковому впровадженню стартапу та визначають фактори перешкоди.

Таблиця 4.6, Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренція	Поява на ринку великої компанії	-Вихід/продаж стартапу; -Спробувати конкурувати сфокусувавшись на певних ключових функціях для найбільш платіжно-спроможної аудиторії; -Залучення додаткових інвестицій
2	Зміна потреб потенційних користувачів	Потенційним користувачам необхідна програма яка виконує інші функції	Предметне вивчення ринку і настроїв користувачів для прогнозування необхідних функцій користувачів
3	Збільшення витрат на розробку та підтримку продукту	Запізніле реагування на потреби користувачів	Кропітке планування необхідних коштів на розробку і підтримку продукту
4	Винайдення нової перспективної технології конкурентної до Honeypot	Нова технологія може спричинити відтік потенційних користувачів	Адаптація продукту до нової технології
5	Збільшення кількості принципів побудови та розгортання Honeypot	Недостатнє покриття та гнучкість рішення	Фокусування на найбільш популярних рішеннях

Були розглянуті фактори загроз для стартапу. Найбільшими загрозами є висока конкуренція у особі великої компанії, яка потенційно може вийти на ринок та зміна потреб потенційних користувачів, можливо їм потрібна програма яка виконує інші функції. Для їх попередження компанії потрібно провести дослідження яке передбачає використання потенційними користувачами програми на безкоштовній основі. У свою чергу це дозволить збільшити клієнтську базу та більше зрозуміти користувачів.

Таблиця 4.7, Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Збільшення кількості та можливостей покупців	Зростання кількості та платоспроможності користувачів. Можлива поява державного фінансування	Розробити додаткові функції для підписки преміум на користування програмою
2	Зниження довіри до конкурентів	З рішеннями конкурентів виникають труднощі або інші негаразди	Рекламна кампанія з дискредитації конкурентів базуючись на відомих недоліках
3	Зменшення витрат на розробку і підтримку програми	Збільшення продуктивності в компанії в цілому	Збільшення якості роботи працівників, підвищення заробітної плати
4	Стандартизація принципів побудови Honeypot	З'являються чіткі стандарти для розробки нових Honeypot	Можливість зробити програму більш гнучкою

Продовження таблиці 4.7			
№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
5	Зростання зацікавленості до технології Honeypot	Все більше компаній починають використовувати honeypot	Визначити які додаткових функцій потребують компанії в першу чергу та почати їх розробку

Були розглянуті основні фактори які можуть спричинити позитивну динаміку для імплементації та прийняття продукту: збільшення кількості і можливостей покупців та недовіри до конкурентів. Ріст зацікавленості до технології Honeypot також може бути дуже сприятливим чинником. Зі свого боку стартап будет реагувати наступним чином: розробляти нові функції, запроваджувати підписку класу преміум, проводити рекламну кампанію з дискредитації конкурентів базуючись на відомих недоліка та збільшення можливостей та гнучкості програми.

Таблиця 4.8, Ступеневий аналіз конкуренції ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
Тип конкуренції: -олігополія	Є декілька фірм які розробляють продукти більш широкого застосування, які можуть включати деякі функції розробляемого продукту	Компанія фокусується на вирішенні проблеми “Виявлення Honeypot” та робить це якісно, дешево та швидше за конкурентів

Продовження таблиці 4.8		
Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
За рівнем конкурентної боротьби: -міжнародний	Усі три компанії конкурента знаходяться у США. Конкурентів із України немає.	Програму слід розробляти на англійській мові для міжнародного ринку
За галузевою ознакою: -внутрішньогалузева	Більшість конкурентів розробляють сканери вразливостей які включаються можливість розпізнавання цілі або рішення які направлені на ідентифікацію потенційних цілей	Програма повинна бути гнучкою для можливої зміни вектора розвитку
Конкуренція за видами товарів: -товарно-видова	Товари різні. Конкурентні продукти націлені на вирішення інших проблем але вони можуть включати деякі функції розробляемого продукту	Програма буде відрізнятись від конкурентів тому що їх рішення сфокусовані на інших проблемах
За характером конкурентних переваг: -націлена	Фокусування та розробка функцій програми лише для вирішенні однієї конкретної проблеми “Виявлення Honeyrot”	Створення програми меншого спектру дії яка буде вирішувати специфічну проблему
За інтенсивністю: -не марочна	Бренди відсутні	-

Був проведений аналіз конкуренції на ринку та визначені тип конкуренції (олігополія) та її вид за ознакою конкурентної боротьби (міжнародна) та

галузевою ознакою (внутрішньогалузева). Також було розглянуто конкуренцію за видами товарів (товарно-видова) та її інтенсивність (не марочна). Були описані дії стартапу задля конкурентоспроможності: дії компанії на початку; фокусування на вирішенні специфічної проблеми, яка не є основною для конкурентів; програма повинна бути якісною, швидкою та дешевшою за конкурентів.

Розглянемо перелік факторів конкурентоспроможності на ринку за допомогою аналізу конкуренції в галузі за М.Портером.

Таблиця 4.9, Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Перелік конкурентів	Бар'єри входження в ринок	Фактори сили постачальників	фактори сили споживачів	Фактори загроз з боку замінників
Висновки	В даний час існує три конкуренти вони є программам и більш широкого застосування , які можуть включати деякі функцію розробляємо го продукту	Немає бар'єрів для входження на ринок	Постачальники відсутні	Важливим для користувача є якість продукту та швидкість роботи продукту	Товари замінники можуть обрати схожі технології до Honeypot та розширити можливості програми

Враховуючи конкурентну ситуацію на ринку можна констатувати, що продукт має перспективи виходу на ринок тому що з наявних конкурентів немає таких які б розробляли такий самий продукт котрий сфокусований на вирішенні зазначеної проблеми. Програма спрощуватиме та пришвидшуватиме виявлення honeypot. Основними сильними сторонами продукту є швидкість, надійність та менша вартість та використання нового підходу до виявлення honeypot який базується на критеріях стійкості.

Таблиця 4.1.1, Обґрунтування факторів конкурентоспроможності

№ п/п	Фактори конкурентоспроможності	Обґрунтування
1	Використання нового підходу до виявлення honeypot який базується на критеріях стійкості	Критерії стійкості створені задля можливості швидкого та зрозумілого оцінювання honeypot
2	Швидкість	Зважаючи на те що конкуренти фокусуються на рішенні інших проблем ми зможемо краще спроектувати програму
3	Надійність	Враховуючи те що програма виконує конкретні функції необхідні для виявлення honeypot. Потенційно можна зробити більш якісну та надійну програму.
4	Менша вартість	Через те що програма буде вирішувати менший спектр проблем ніж програми конкурентів можна запропонувати кращу ціну для потенційних користувачів

Були виділені фактори конкурентоспроможності за допомогою яких стартап-проект планує перемогти конкурентів: використання нового підходу до виявлення honeypot який базується на критеріях стійкості, швидкість, надійність та менша вартість програми.

На базі виділених факторів визначаються сильні та слабкі сторони стартапу.

Таблиця 4.1.2, Порівняльний аналіз сильних та слабких сторін страп-проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів конкурентів						
			-3	-2	-1	0	1	2	3
1	Використання нового підходу до виявлення honeypot який базується на критеріях стійкості	20	+						
2	Швидкість, надійність, менша вартість	15			+				

Були визначені сильні та слабкі сторони стартапу у порівнянні з запропонованими рішеннями конкурентів. Ключовою сильною стороною є використання нового підходу до виявлення honeypot який базується на критеріях стійкості.

Проведемо SWOT-аналіз результати якого містяться у таблиці 4.1.3

Таблиця 4.1.4, SWOT аналіз стартап-проекту

Сильні сторони: використання нового підходу до виявлення який базується на критеріях стійкості, швидкість, надійність, менша вартість	Слабкі сторони: потрібен час для прийняття алгоритму, також необхідно виявляти недоліки та покращувати його дію
Можливості: так як наразі на ринку немає продукту який би вирішував цю специфічну проблему є можливість стати першою програмою і захопити певну частину ринку.	Загрози: висока конкуренція, зміна потенційних потреб користувачів, збільшення витрат на розробку і підтримку програми

Маємо такі характерні сильні і слабкі сторони відносно виведення на ринок стартапу: сильні - використання нового підходу до виявлення який базується на критеріях стійкості, швидкість, надійність, менша вартість; слабкі - потрібен час для прийняття алгоритму, також необхідно виявляти недоліки та покращувати його дію. Ринкові можливості стартапу відносно зовнішнього оточення виділяються наступними можливостями і загрозами: можливості - стати першою програмою і захопити певну частину ринку; загрози - висока конкуренція, зміна потенційних потреб користувачів, збільшення витрат на розробку і підтримку програми.

На базі проведеного SWOT-аналізу розробляються альтернативні можливості ринкового впровадження.

Таблиця 4.1.5, Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива	Ймовірність отримання ресурсів	Сроки реалізації
1	Створення програми за допомогою мови програмування Python	85%	6 місяців
2	Створення програми за допомогою мови програмування Ruby	75%	7 місяців
3	Створення програми за допомогою мови програмування C++	25%	10 місяців

Було обрано альтернативу ринкового впровадження, а саме розробка програми мовою програмування Python тому що за рахунок меншого необхідного часу реалізації виділення коштів є більш вигідним.

4.4 Розроблення ринкової стратегії проекту

Першим кроком до формування ринкової стратегії є визначення стратегії охоплення ринку а саме цільових груп кінцевих користувачів. Вони були проаналізовані відштовхуючись від бачення концепції стартап-проекту та наведенні у таблиці 4.1.6.

Таблиця 4.1.6, Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприяти продукту	Орієнтовний попит в межах цільової групи	Інтенсивні сть конкуренц ії в сегменті	Простота входу у сегмент
1	Розробники Honeyrot	Отримання даних про користуванн я продуктом та можливість його покращення	Високий	Існує 3 конкурент і які розробля ють продукти які мають кілька функцію розробляє мого продукту	Проста
2	Компанії	Отримання даних про користуванн я продуктом та можливість його покращення	Високий		Середня
3	Спеціалісти з кібербезпеки	Отримання даних про користуванн я продуктом та можливість його покращення	Середній		Середня
Було обрано: Розробників Honeyrot та компанії.					

Було вирішено обрати розробників Honeypot через найлегшу можливість входу у сегмент та компанії високий попит серед цільової групи.

Таблиця 4.1.7, Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Створення програми мовою програмування Python яка базуватиметься на критеріях стійкості до розпізнавання	Ринкове позиціювання	Швидкість, надійність, менша вартість	Диференціація

Було обрана альтернатива розвитку проекту, - створення програми мовою програмування Python яка базуватиметься на критеріях стійкості до розпізнавання та визначена стратегія розвитку продукту - диференціація, яка базується на позиціюванні тому що основними перевагами продукту є швидкість, надійність, менша вартість.

Далі, необхідно визначити базову стратегію конкурентної поведінки, яка наведена у таблиці 4.1.8.

Таблиця 4.1.9, Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект першопрохідцем на ринку	Чи буде компанія шукати нових споживачів або забирати існуючих у конкурентів	Чи буду компанія копіювати основні характеристики	Стратегія конкурентної поведінки
1	Так	Так	Компанія буде звертати увагу та аналізувати на модель ведення бізнесу, успіхи, невдачі та інтерфейси конкурентів	Зайняття якогого більшої частини ринку до появи прямих конкурентів

Було обрано базову стратегію конкурентної поведінки, зайняття якомога більшої частини ринку до появи прямих конкурентів, бо поки їх немає така існує можливість. Також було визначено, що компанія буде звертати увагу та аналізувати модель ведення бізнесу, успіхи, невдачі та інтерфейси конкурентів. Детальний аналіз яких допоможе краще зрозуміти ринок на якому працюють продукти які мають деякі схожі функції та можливе ставлення до них потенційних користувачів.

Таблиця 4.2.1, Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможності і позиції стартап-проекту	Вибір асоціацій які мають сформувати комплексну позицію власного проекту
1	Програма повинна чітко виконувати основну функцію “Виявлення Honeypot” базуючись на критеріях стійкості	Диференціація	“Виявлення Honeypot” базуючись на критеріях стійкості це основна функція на основі якої базується продукт	Передова технологія
2	Швидкість	Диференціація	Програма виконує меншу кількість функцій ніж у конкурентів тому повинен бути швидшим за конкурентів	Максимальна швидкодія
3	Надійність	Диференціація	Надійність продукту дуже важлива для потенційних клієнтів особливо на етапі формування та виходу на ринок	Надійність
4	Менша вартість	Диференціація	На розробку та підтримку продукту з меншу кількість функцій потрібно менше ресурсів тому продукт може бути дешевшим	Менша вартість та якість

Були виділенні ключові вимоги цільової аудиторії до товару: чітке виконання зазначених функцій, швидкість, надійність та менша вартість. Також була обрана стратегія розвитку, ключові позиції конкурентоспроможності та набір асоціацій, які повинні сформуватися у користувачів.

4.5 Розроблення маркетингової програми стартап-проекту

Для створення маркетингової програми стартапу, перш за все, потрібно сформулювати маркетингову концепцію продукту.

Таблиця 4.2.2, Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода яку пропонує товар	Ключові переваги перед конкурентами
1	Автоматизація виявлення Honeypot	Автоматизоване рішення проблеми виявлення Honeypot	Програма автоматизує процес виявлення Honeypot існуючу вирішуючи проблему
2	Швидкість	Швидкість роботи програми	Програма працює швидше за конкурентів
3	Надійність	Надійність роботи програми	Програма викликає довіру під час використання
4	Менша вартість	Вартість програми	Продукт є дешевшим за конкурентів через фокусування на конкретній проблемі

Перейдемо до розробки трирівневої маркетингової моделі товару яка наведена у таблиці 4.5.2 .

Таблиця 4.2.3, Опис трьох рівнів моделі товару

Рівні товару	Сутність та складова		
Товар за задумом	Товар автоматизує процес розпізнавання Honeypot		
Товар у реальному виконанні	Властивості	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Автоматизація виявлення Honeypot 2. Швидкість 3. Надійність 4. Менша вартість	Нм	Тх
	Якість: розробка програми буде виконана за концепцією SDL		
	Пакування відсутнє		
	HoneyDetection		
Товар із підкріпленням	2-х недільне безкоштовне користування програмою		
	Підтримка користувачів		
За рахунок чого потенційний товар буде захищено від копіювання: патент			

Отже були визначені три рівні моделі проекту: він буде виконаний у вигляді програми та допомагатиме автоматично виявляти Honeypot. Основними характеристиками є: швидкість, надійність, менша вартість та безпека розробляємої програми. Програму буде захищено від копіювання за допомогою патенту.

Далі, необхідно визначити цінову межу, вона буде визначена за допомогою експертного метода.

Таблиця 4.2.4, Визначення між встановлення ціни

№ п/ п	Рівень цін на товари-замінники, грн/рік	Рівень цін на товари-аналоги, грн/рік	Рівень доходів цільової групи споживачів, грн/рік	Верхня та нижня межі встановлення ціни на товар/послугу, грн/рік
1	60000	50000	400000	20000, 30000

Були визначені межі можливих цін за користування програмою, рівень цін товарів-аналогів та доходів споживачів. Верхньою та нижньою межами є 20000, 30000 грн/рік відповідно.

Перейдемо до оптимальної системи збуту у межах якої можуть прийматися потенційні рішення.

Таблиця 4.2.5, Формування системи збуту

№ п/ п	Специфіка закупівельної поведінки клієнтів	Функції збуту, яка має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Клієнти купують річну підписку на користування програмою	Продаж	За власними каналами	За власними каналами

Було сформовано систему збуту у формі підписки на рік. Збут буде виконуватися за власними каналами.

Наступним кроком є розробка концепції маркетингових комунікацій стартапу.

Таблиця 4.2.6, Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Завантаження програми з офіційного сайту продукту	Інтернет	Швидкість, надійність, менша вартість	Показати як продукт вирішує проблему користувачів та в чому є переваги над конкурентами	Ролик з демонстрацією роботи продукту

Були розроблені концепції можливих маркетингових комунікацій.

Завантаження для кожної платформи буде виконуватися з офіційного сайту, канал комунікації - інтернет. Рекламне повідомлення повинно бути у формі демонстраційного відео та показати вирішення проблеми та переваги потенційному користувачеві.

4.7 Висновки до розділу 4

У розділі була описана ідея, концепція та основна цінність стартап-проекту. Програма націлена на розробників, спеціалістів з кібербезпеки та компанії які прагнуть покращити використовувані нею рішення щодо збору інформації і аналізу проведення атак за допомогою Honeypot.

У рамках розділу були визначені сильні, нейтральні та слабкі сторони та властивості стартапу задля формування його конкурентоспроможності. Продукт буде створений у вигляді програми та розроблений за допомогою мови програмування Python, яка є безкоштовною та з нею знайомі члени проекту. Також було виконано: ступеневий аналіз конкуренції, SWOT аналіз, менеджмент ризиків.

У відповідності до проведених досліджень можна констатувати, що є можливість і перспективи комерціалізації продукту. Бар'єри входження - низькі. Стартап-проект має переваги над існуючими рішеннями: швидкість, надійність, менша вартість.

Для успішного впровадження стартап-проекту необхідно розробити програму для виявлення Honeypot базуючись на критеріях стійкості до виявлення.

Беручи до уваги отримані результати, можна констатувати, що стартап-проект є актуальним та перспективним.

ВИСНОВОК

Під час виконання даної магістерської дисертації було розглянуто концепцію Honeypot, його історію, значення, різні методи реалізацію та можливість їх виявлення. Honeypot надає можливість збирати та аналізувати інформацію про дії зловмисників, які намагалися атакувати систему.

Було розглянуто та проаналізовано вже існуючі рішення до можливості виявлення Honeypot. З розглянутих рішень було обрано три реалізації honeypot, які було вирішено розгорнути терміном не менше 5 днів. Після закінчення дослідження були проаналізовані отримані результати, а саме: кількість атак, кількість унікальних IP-адрес, кількість завантажених шкідливих файлів та кількість користувачів які намагалися встановити що вони знаходяться у Honeypot.

Базуючись на отриманих результатах було розроблено програму для виявлення Honeypot Cowrie , який зазнав найбільшу кількість атак під час дослідження. Також було проведене незалежне випробування дієвості програми на незалежно розгорнутому Honeypot Cowrie знайденому за допомогою пошукової системи Shodan.

Було проведене невелике дослідження на предмет наявності серед загально доступних приманок таких, які розгорнуті на базі звичайного обладнання без присутності віртуальної складової та можливість їх виявлення за певними ознаками. Було встановлено, що частіше, дійсно використовується застаріле обладнання для розгортання приманок, які не потребують використання віртуальної складової під час розгортання. Також було підтверджено можливість виявлення honeypot за ознакою використання застарілого обладнання.

На базі даної роботи та загальновідомих методів виявлення Honeypot були розроблені критерії стійкості до виявлення Honeypot. За створеними критеріями

стійкості був проведений аналіз основних типів honeypot, які розглядалися у першому розділі.

Був розроблений детальний план створення стартап-проекту та досліджені основні аспекти виходу на ринок програми для виявлення Honeypot. Розроблена програма є доцільною для розробників Honeypot, спеціалістів з кібербезпеки та компаній, які використовують приманки у якості інструменту для збору та аналізу дій злоумисників. Було визначено слабкі, нейтральні та сильні сторони потенційного продукту для формування його конкурентоспроможності. Для реалізації програми було вирішено використовувати мову програмування Python та розроблені критерії стійкості Honeypot до розпізнавання. В процесі дослідження конкурентоспроможності стартапу був проведений ступеневий аналіз конкуренції на потенційному ринку, SWOT аналіз та виконаний менеджмент можливих ризиків.

Базуючись на результатах проведених досліджень можна констатувати, що існує можливість потенційної комерціалізації проекту. Існує і перспектива впровадження продукту. Враховуючи те, що немає бар'єрів для виходу на ринок і продукт має переваги над конкурентами: швидкість, надійність, менша вартість. Проаналізувавши результати створення стартап-проекту, можна зробити висновок, що створення стартапу та подальший його розвиток є доцільним.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Corey, J. (2005). Advanced Honey Pot Identification and Exploitation.
2. Dike, J. (2005). UML as a honeypot.
3. Holz, T. Raynal, F. (2005). Defeating Honeypots: System Issues, Part 1
4. Holz, T. Raynal, F. (2005). Defeating Honeypots: System Issues, Part 2
5. Oudot, L. Holz, T. (2004). Defeating Honeypots: Network Issues, Part 1
6. Oudot, L. Holz, T. (2004). Defeating Honeypots: Network Issues, Part 2
7. NG, Chee Keong, Pan, Lei, Xiang, Yang (2018). Honeypot Frameworks and Their Applications: A New Framework
8. R. Marlaan Rush (2014). The Honeypot Chronicles 2

КОД ПРОГРАМИ ДЛЯ ВИЯВЛЕННЯ HONEYPOT COWRIE

```
import platform
```

```
import argparse
```

```
import socket
```

```
import sys
```

```
import time
```

```
DEF_COWRIE_BANNERS=["SSH-2.0-OpenSSH_5.9", "SSH-1.99-Sun_SSH_1.1",  
"SSH-2.0-OpenSSH_4.6",
```

```
"SSH-2.0-OpenSSH_5.1p1 Debian-5", "SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu6",  
"SSH-1.99-OpenSSH_4.3", "SSH-1.99-OpenSSH_4.7", "SSH-2.0-OpenSSH_4.2p1  
Debian-7ubuntu3.1", "SSH-2.0-OpenSSH_4.3",
```

```
"SSH-2.0-OpenSSH_5.1p1 Debian-5", "SSH-2.0-OpenSSH_5.1p1  
FreeBSD-20080901", "SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu5",  
"SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu7", "SSH-2.0-OpenSSH_5.5p1  
Debian-6", "SSH-2.0-OpenSSH_5.5p1 Debian-6+squeeze1", "SSH-2.0-OpenSSH_5.5p1  
Debian-6+squeeze2", "SSH-2.0-OpenSSH_5.8p2_hpn13v11  
FreeBSD-20110503", "SSH-2.0-OpenSSH_5.9p1 Debian-5ubuntu1",  
"SSH-2.0-OpenSSH_6.0p1 Debian-4+deb7u2", "SSH-2.0-OpenSSH_6.0p1  
Debian-4+deb7u2"]
```

```
DEF_BANNER = "SSH-2.0-OpenSSH_"
```

```
DEF_PORT = 22
```

```
VERB = True
```

```
ERR = -1
```

```
def getBanner(ServerBanner):
```

```
    banner = ServerBanner.decode('utf-8').strip()
```

```
    if banner in DEF_COWRIE_BANNERS:
```

```
        print("Note: The banner of this server used in the Cowrie default list")
```

```
    return DEF_BANNER in banner
```

```
def conToSSH(host, port):
```

```
    try:
```

```
        socket.setdefaulttimeout(5)
```

```
        sockfd = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
        sockfd.connect((host, port))
```

```
        banner_one = sockfd.recv(1024)
```

```
        if getBanner(banner_one):
```

```
            if VERB:
```

```
                print("[+] The host %s:%d positions itself as the OpenSSH. Let's  
continuing.." % (host, port))
```

```
            else:
```

```
                print("[!] The host %s:%d does not positions itself as OpenSSH. Exit.." %  
% (host, port))
```

```
    return False
```

```

except Exception as err:

    print("[!] Unable connect to the host %s on port %d: %s" % (host, port,
str(err)))

    return False

return sockfd

def probIncorrectVers(sockfd):

    try:

        sockfd.sendall('SSH-1337\n'.encode('utf-8'))

    except Exception as err:

        print("[!!] The error appeared during sending the first probe: %s" % str(err))

    response = sockfd.recv(1024)

    sockfd.close()

    if VERB:

        print(response)

    if b"bad_version" in response:

        if VERB:

            print("[+] Got the incorrect version in the first probe. It could be a
HoneyPot!\n")

        return True

    else:

```



```

        return False

def probPackets(sockfd):

    try:

        sockfd.sendall("SSH-2.0-OpenSSH\n\n\n\n\n\n\n\n\n\n".encode('utf-8'))

    except Exception as err:

        print("[!!] The Error appeared during sending the second probe: %s" %
str(err))

    response = sockfd.recv(1024)

    sockfd.close()

    if b"corrupt" in response or b"mismatch" in response:

        if VERB:

            print("[+] Got the 'packet corrupt'/'protocol mismatch' in second probe
response . Should be a HoneyPot!\n")

            return True

        else:

            return False

def probeDuplicate(sockfd):

    try:

        sockfd.sendall("SSH-2.0-OpenSSH_6.0p1
Debian-4+deb7u2\nSSH-2.0-OpenSSH_6.0p1 Debian-4+deb7u2\n".encode('utf-8'))

    except Exception as err:

```

```

    print("[!!] The Error appaered during sending the third probe: %s" % str(err))

response = sockfd.recv(1024)

sockfd.close()

if b"corrupt" in response or b"mismatch" in response:

    if VERB:

        print("[+] Got the 'packet corrupt'/protocol mismatch' in third probe
response. Could be a HoneyPot!\n")

        return True

    else:

        return False

def detectCowrie(host, port):

    res = 0

    print("[1] The first detection method of Cowrie Honeypot - The incorrect
version used")

    sockfd = conToSSH(host, port)

    if sockfd:

        if probIncorrectVers(sockfd):

            res += 1

    else:

        print("The socket error appeared, first method [1]")

    sys.exit(ERR)

```

```
print("[2] The second detection method of Cowrie Honeypot - The spacer
used")
```

```
sockfd = conToSSH(host, port)
```

```
if sockfd:
```

```
    if probPackets(sockfd):
```

```
        res += 1
```

```
else:
```

```
    print("The socket error appeared, second method [2]")
```

```
    sys.exit(ERR)
```

```
print("[3] The third detection method of Cowrie Honeypot - The Double banner
used")
```

```
sockfd = conToSSH(host, port)
```

```
if sockfd:
```

```
    if probeDuplicate(sockfd):
```

```
        res += 1
```

```
else:
```

```
    print("The socket error appeared, second method [3]")
```

```
    sys.exit(ERR)
```

```
return res
```

```
def main():
```

```

if len(sys.argv) >= 1:

    host = sys.argv[1]

    port = int(sys.argv[2])

    res = detectCowrie(host, port)

    if res >= 2:

        print("\t\t\t[+] The Cowrie Honeypot detect on host: %s port: %d " % (host,
port))

    elif res == 1:

        print("\t\t\t[+/-] The host %s on port: %d could be a Cowrie honeypot or not
properly configurated OpenSSH" % (

            host, port))

    elif res == 0:

        print("\t\t\t[-] The host %s on port %d not a Honeypot!." % (host, port))


if __name__ == '__main__':

    main()

```