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

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи)

Магістр
(освітній ступінь (освітньо-кваліфікаційний рівень))

на тему: **Методи розробки адаптивних контейнерних приманок (honeypot) для моніторингу кіберінцидентів**

Виконав: студент (ка) 6 курсу, групи СБм-61
спеціальності (напряму підготовки)
125 - кібербезпека
(шифр і назва спеціальності (напряму підготовки))

(підпис) Бельма А. В. (прізвище та ініціали)

Керівник (підпис) Кареліна О. В. (прізвище та ініціали)

Нормоконтроль (підпис) Кареліна О. В. (прізвище та ініціали)

Рецензент (підпис) Баран І. О. (прізвище та ініціали)

м. Тернопіль – 2019

АНОТАЦІЯ

Дипломна робота за другим (магістерським) рівнем вищої освіти на тему: «Методи розробки адаптивних контейнерних приманок (honeypots) для моніторингу кіберінцидентів» містить 179 сторінок, 8 таблиць, 35 рисунків, 1 додаток. Перелік посилань нараховує 114 найменувань.

Метою роботи є проектування та розроблення високо-розгорнутої системи моніторингу кіберінцидентів для консолідації інформації про загрози, зібраної з мережі приманок, для забезпечення підвищеної безпеки засобами моніторингу інцидентів.

Об'єктом дослідження є модель Honeypot для захисту інформаційних ресурсів.

Предметом дослідження є методи та засоби організації систем збору та моніторингу інформації з приманок.

Методика дослідження. У дипломній роботі було використано такі методи: аналізу, синтезу, узагальнення результатів дослідження, економічний, статистичний, порівняльний та ін.

Результати роботи: у результаті підготовки дипломної роботи розроблено повно мережеву систему приманок для забезпечення активного захисту мережі в сучасних IT-інфраструктурах, упаковану як єдиний розгорнутий блок. Також запропоновано методи адаптації конструкції приманки для більш ефективного залучення атак.

Ключові слова: приманки, мережа приманок, віртуалізація, емуляція, контейнеризація, захист інформації, збір інформації, відмова в обслуговуванні, виявлення атак, аналіз інциденту, моніторинг, ядро Лінукс (Linux), гіпервізор, платформа як послуга (Platform-as-Service), програма як послуга (Software-as-Service), кейлоггінг, брандмауер.

ANNOTATION

Master’s thesis: «Methods of development of adaptive container honeypots for cyber incident monitoring» includes 179 pages, 8 tables, 35 drawings, 4 listings, 1 appendix. The bibliography list consists of 114 items.

The purpose of the work is design and development of a highly-deployed cyber incident monitoring system to consolidate threat information collected from the Honeynet to provide enhanced security through incident monitoring.

The object of the study is a Honeypot model for information resources protection.

The subject of the study is methods of organizing systems for collecting and monitoring honeypot information.

The research methods: methods of analysis and synthesis, method of generalization of research results, economic, statistical, graphical, comparative, etc.

The results of the study: as a result of the preparation of the thesis, a full-network Honeypot system, packaged as a single deployed unit, has been developed to provide active network security in today's IT infrastructures. Methods of adaptation of honeypot design to more effectively entice attacks are also described.

Keywords: honeypot, honeynet, virtualization, emulation, containerization, information security, information gathering, denial of service, incident detection, monitoring, Linux kernel, hypervisor, Platform-as-Service, Infrastructure-as-Service, keylogging, firewall.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API Application Programming Interface – інтерфейс прикладного програмування

AWS (Amazon Web Services) – веб-сервіси Amazon

C+C Command and Control - управління і контроль

CLI (Command Line Interface) – інтерфейс командного рядка

DoS (Denial of Service) – атака на відмову в обслуговуванні

DDoS (Distributed Denial of Service) – розподілена атака на відмову в обслуговуванні

DMZ (De-Militarised Zone) – демілітаризована зона

DNS (Domain Name System) – Система доменних імен

DPI (Deep-Packet Inspection) – Глибока перевірка пакетів

EC2 (Elastic Compute Cloud) - Еластична обчислювальна хмара

ELK (Elasticsearch-Logstash-Kibana) – інфраструктурний програмний засіб, розроблений компанією Elastic, який складається з декількох компонентів

FTP (File Transfer Protocol) – протокол передачі файлів

HTTP (Hypertext Transfer Protocol) – протокол передачі гіпертексту

IaaS (Infrastructure as a Service) – Інфраструктура як послуга

ICMP (Internet Control Message Protocol) - міжмережевий протокол керуючих повідомлень

IDS (Intrusion Detection System) – Система виявлення вторгнень

IETF (Internet Engineering Task Force) - Відкрите міжнародне співтовариство проектувальників, учених, мережевих операторів і провайдерів

IoT (Internet of Things) – Інтернет речей

IP (Internet Protocol) – між мережевий протокол

IT (Information Technology) – Інформаційна технологія

JSON (JavaScript Object Notation) – текстовий формат обміну даними між комп’ютерами

LPWAN (Low-Powered Wide-Area Network) – енергоефективна мережа широкого радіуса дії

LXC (Linux Container) – система віртуалізації на рівні операційної системи для запуску декількох ізольованих примірників операційної системи Linux на одному вузлі

MHN (Modern Honey Network) – система управління приманками

NAT (Network Address Translation) – механізм перетворення мережних адрес

NHS (National Health Service, UK) – Національний центр здоров’я Великобританії

NTP Network Time Protocol - мережевий протокол синхронізації внутрішнього годинника комп'ютера

ОС (Operating System) – операційна система

PaaS Platform as a Service – платформа як сервіс

PSAD (Port Scan Attack Detection) – система виявлення сканування портів

RAM (Random Access Memory) – оперативна пам’ять

RCE (Remote Code Execution) – віддалене виконання коду

RSA (Rivest, Shamir, and Adelman) – криптографічний алгоритм з відритим ключем, що базується на обчислювальній складності задачі факторизації великих чисел

SANS (System Administration, Networking, and Security Institute) - Інститут, який займається дослідженнями та освітніми програмами в галузі інформаційної безпеки, системного адміністрування, аудиту

SCADA (Supervisory Control and Data Acquisition) – програмний пакет, призначений для розробки або забезпечення роботи в реальному часі систем збору, обробки, зображення та архівування інформації про об'єкт моніторингу або управління

SHA (Secure Hash Algorithm) - алгоритм криптографічного хешування

SMB (Server Message Block) – протокол прикладного рівня, який використовується для надання розділеного доступу до файлів, принтерів,

послідовних портів передачі даних та іншої взаємодіє між вузлами в комп’ютерній мережі

SMTP (Simple Mail Transfer Protocol) – комунікаційний протокол для пересилання електронної пошти

SNMP (Simple Network Management Protocol) – протокол керування мережами зв’язку на основі архітектури TCP/IP

SSH (Secure SHell) – мережевий протокол рівня за стосунків, що дозволяє проводити віддалене управління комп’ютером і тунелювання TCP-з’єднань

SSL (Secure Sockets Layer) - криптографічний протокол, який забезпечує встановлення безпечного з’єднання між клієнтом і сервером

TCP (Transmission Control Protocol) – протокол передачі даних

TFTP (Trivial File Transfer Protocol) – тривіальний протокол передачі файлів

TLS (Transport Layer Security) – криптографічний протокол, що надає можливості безпечної передачі даних в Інтернет для навігації, отримання пошти, спілкування, обміну файлами, тощо.

VM (Virtual Machine) – віртуальна машина

WSN (Wireless Sensor Network) – бездротова сенсорна мережа

ВСТУП

Актуальність дослідження. На сьогодні до Інтернету під’єднано більше систем і пристроїв, ніж будь-коли раніше. Цією величезною кількістю пристроїв все частіше користуються люди з обмеженими технічними знаннями й ще меншою обізнаністю про загрози їх безпеки. З розширенням масштабів погано налаштованих заходів безпеки в системах, сучасний Інтернет стає ідеальним для зловмисників майданчиком з нескінченною кількістю пристроїв, які можуть бути зламані, експлуатовані та використані в безчесних цілях. Тому особливо актуальним є питання виявлення та протидії як старим, так і найновішим, ще невідомим типам та методам зламу, вірусах, шкідливому програмному забезпеченню та іншій зловмисній діяльності.

Вирішення цього складного завдання покладено на системи-приманки (англ. – Honeypots) – привабливі для атаки об’єкти, що функціонують на основі спеціалізованого програмного забезпечення (ПЗ) та мають можливості реєструвати процедури зламу, виявляти та аналізувати мережні атаки. Серед науковців, які розв’язували окремі проблеми в цій області, можна назвати таких відомих вчених як С. Даулінг, М. Антонакакіс, І. Мокубе, М. Наврокі, Е. Васіломанолакіс, Т. Баррон, Н. Провос, П. Пісарчік та інші. Проте значна кількість проблем ефективного використання приманок, попри зусилля багатьох розробників, досі не вирішена. Зокрема, існуючі дослідження не приділяють достатньо уваги тому, як спонукати зловмисників взаємодіяти з приманками, натомість зациклюючись на самій взаємодії.

Мета і завдання дослідження. Метою дипломної роботи є дослідження та розроблення методів побудови контейнерної повно-інтерактивної мережі приманок з активним моніторингом, спрямованої на забезпечення стійкості до впливів зловмисника, прихованості й ізольованості механізмів збору та обробки інформації. Реалізація мети дослідження обумовила поставлення та розв’язання таких завдань:

1. Аналіз методів проектування приманок і візуалізації їх даних засобами стеку ELK.

2. Проектування адаптивної приманки для заохочення взаємодії з нею.

3. Розроблення інтерактивної мережі приманок з використанням контейнерного середовища.

Об’єктом дослідження є процес отримання та аналізу інформації з систем, що виконують роль приманок, з метою вивчення діяльності зловмисника під час зламу.

Предметом дослідження є методи та засоби організації систем збору та моніторингу інформації з приманок.

Методи дослідження. В процесі дослідження використано методи побудови обчислювальної мережі, контейнеризації, TCP/IP, Honeypot, Honeynet, візуалізація за допомогою стеку ELK. Також використовувались загальнонаукові методи пізнання: порівнювання, системний аналіз, моделювання.

Інформаційною основою дослідження є праці закордонних вчених з питань розробки та ефективного використання приманок.

Наукова новизна роботи:

1. Вперше розроблено повно-мережеву систему інтерактивних приманок з активною системою моніторингу як єдиний розгорнутий блок, який можна розмістити в Linux-системах.

2. Вперше розглянуто адаптацію конструкції приманки для більш активного залучення атак.

Практичне значення дослідження полягає у можливості застосування розробленої системи в діяльності об’єктів критичних інфраструктур для активної протидії кіберзагрозам.

Апробація. Окремі результати роботи обговорювались та були схвалені на VII науково-технічній конференції «Інформаційні моделі, системи та технології» (Тернопіль, 2019).

Структура роботи. Дипломна робота складається із вступу, 7 розділів, висновків, бібліографії із 114 найменувань. Робота містить 35 рисунків, 8 таблиць, 4 лістинги і 1 додаток. Обсяг основного тексту становить 103

сторінки, бібліографія 12 сторінок. Загальний обсяг дипломної роботи складає 179 сторінок.

1 МІСЦЕ ТА ЗНАЧЕННЯ ПРИМАНОК HONEYPOT В СУЧАСНОМУ КІБЕРПРОСТОРІ

1.1 Сучасний ландшафт кіберзагроз

У 2019 році ландшафт кіберзагроз продовжує змінюватися та трансформуватися. Кібер-інциденти не локалізуються, а натомість відбуваються в глобальному масштабі, їх охоплення поширюється на всі фізичні кордони та юрисдикції. Недавнім прикладом є подія червня 2017 року, коли атака російських військових з використанням шкідливого програмного забезпечення NotPetya призвела до зупинки значної частини української економіки [1]. Зростання різноманітності та серйозності атак призвело до формування в країнах по всьому світу нових робочих груп з питань безпеки з метою визначення способів захисту свого суспільства від зростаючих кіберзагроз. До прикладу, в лютому 2018 року генеральний прокурор США Джефф Сессіонс оголосив про створення нової кіберцифрової цільової групи у складі Міністерства юстиції США [2].

1.1.1 Кібератаки

Кібератаки продовжують зростати в розповсюдженості та різноманітності завдяки постійній концепції та впровадженню нових технологій. Часто успіх цих атак менше пов'язаний з майстерністю тих, хто за ними стоїть, і більше з недостатньою обізнаністю та увагою, яку ставить їх жертва. Багато успішних кібератак не були особливо складними й часто виникали внаслідок експлуатації відомих уразливостей, які не було усунено.

Загалом атаки можна класифікувати як навмисні чи ненавмисні.

1) Навмисні атаки

Кібер-атаки можна вважати навмисними, якщо жертва була мішенню нападника [3]. Найбільш поширеними мотивації до навмисних кібератак є:

- Кіберзлочинність
- Активізм та тероризм
- Атаки держав та кібервійни

2) Ненавмисні атаки

Кібератаки можуть бути класифіковані як ненавмисні, якщо жертва не була спеціально піддана нападу, але випадково опинилася жертвою атаки. Найпоширеніша мотивація ненавмисної кібератаки – соціальна інженерія за допомогою таких схем, як фішинг-шахрайство з електронною поштою.

Фази, залучені в розвиток кібератаки, залежать від безлічі факторів, включаючи мотивацію атаки, тип атакуючого і характер атаки.

В контексті цього дослідження зловмисниками є ті, хто навмисно атакують IT-системи та інфраструктури. У галузі безпеки широко використовуються різні інші терміни для позначення цих людей, включаючи «поганих акторів» та «чорних капелюшків».

На сьогодні навчитись атакувати IT-системи досить легко. В Інтернеті доступні нескінченні ресурси, набори інструментів і відкриті спільноти, що дозволяють людині успішно виконувати експлойти в будь-якому масштабі.

Зловмисники, як особи, які вчиняють зловмисну діяльність, зазвичай прагнуть приховати свою справжню особистість. Існує безліч різних класифікацій атакуючих, деякі з них описаних нижче:

- Досвідчені зловмисники

До досвідчених відносяться зловмисники з високими навичками технічного рівня, здатні виявити нові категорії вразливостей та створювати потужні набори інструментів для атаки. Загалом, від них найважче захищатись через їхні значні знання та вміння.

- Скрипт-кідді

Так звані «скрипт-кідді» ініціюють атаки за допомогою шкідливих програм та наборів інструментів, створених іншими зловмисниками, тобто не являються їх розробниками.

- Хактивісти (Hacktivists)

Ці зловмисники здійснюють хактівістичні атаки. Вони мотивовані ідеєю, яку вони прагнуть просувати та пропагувати, і можуть охоплювати терористів та організації, які фінансуються державою. Вони часто прагнуть завдати шкоди іншій стороні. Добре відома група хактівістів, яка часто потрапляє в заголовки газет, є Anonymous [5].

- Інсайдери

Інсайдери є однією з найбільших загроз безпеці організацій, оскільки вони є довіреними особами всередині її мережі.

- Боти

Боти це спеціальні програми, які проводять автоматичні та/або з заданим розкладом атаки, виконуючи шкідливі дії/інструкції. Група пристроїв, керованих такими шкідливими програмами, називається ботнетом. Шкідливе ПО, що використовується для зараження цих пристроїв, поширюється від пристрою до пристрою, прагнучи заразити якомога більше. При компрометації використовуються обчислювальні ресурси пристроїв, що дозволяє так званому ботмайстру – людині, яка керує ботнетом, координувати пристрої для виконання певної функцій як колективної групи.

При формуванні атаки на систему існує 5 основних етапів, які зазвичай використовуються [4]:

1) Розвідка

Розвідка містить в собі дослідження мети, і, як правило, виконується шляхом збору інформації, яка доступна у відкритому доступі і не взаємодіє безпосередньо з системами, пов'язаними з метою. Цей вид збору інформації може також містити дослідження таких елементів систем, як використовувані технології, версії ПЗ тощо, дозволяючи зловмисникові дослідити обрані цілі в системі.

2) Перерахунок

На цьому етапі зловмисник використовує інформацію, зібрану на етапі розвідки, щоб почати зондування цільових систем.

3) Проникнення

Цей етап охоплює використання будь-яких вразливостей, виявлених зловмисником на етапах розвідки і перерахунку, з метою отримання доступу до цільової системи. Як тільки зловмисник отримає доступ, він спробує зберегти точку опори в системі за допомогою таких заходів, як підвищення привілеїв і експлуатації інших підключених систем, з наміром підтримувати «Командування і контроль» (C+C). При автоматичній атаці це дозволяє скомпрометованому пристрої обмінюватись даними між зовнішніми системами C+C для отримання команд і подальшого впливу на них.

4) Витік даних

Після злому цільової системи зловмисник спробує зберегти до неї доступ, щоб не втратити контроль в майбутньому. Задля цього, в систему встановлюються так звані механізми бекдору.

5. Очищення даних про втручання

Як вже було наголошено, зловмисники майже завжди прагнуть залишатися анонімними: в їх інтересах залишатися непоміченими після компрометації системи, щоб експлуатація системи не була виявлена. Щоб забезпечити це, перед відключенням від скомпрометованої системи зловмисник спробує видалити усі докази своєї присутності в системі.

1.1.2 IoT Пристрої та їх безпека

Інтернет речей (IoT) – це концепція, що має багато аспектів, оскільки вона охоплює величезний спектр технологій, стандартів і послуг. В цілому, система IoT – це сукупність інтелектуальних пристроїв, що працюють на спільній основі для досягнення спільної мети [8].

В останні кілька років IoT захопив світ штурмом, скоротивши розрив між фізичним світом та Інтернетом. Зв'язок швидко впроваджується в критично важливі системи та процеси: в галузях і секторах, включаючи сільське господарство, енергетику, охорону здоров'я і транспорт. Всі революційні

перетворення стали частиною того, що було названо «Четверта промислова революція».

Згідно зі статистикою, опублікованою Statista у 2018 році [25], до 2025 року прогнозується, що у світі буде понад 75 мільярдів IoT-пристроїв: в кілька разів більше, ніж його населення [10]. На рисунку 1.1 показана їх прогнозована експоненціальна тенденція зростання за період з 2015 року. Ці пристрої охоплюють все – від датчиків навколишнього середовища до переносних технологій і компонентів автоматизації процесів, і грають все більш важливу роль в житті всіх членів суспільства. Для однієї системи IoT датчики можуть використовуватися для збору інформації про конкретне середовище в конфігурації бездротової сенсорної мережі (WSN) при передачі інформації в віддалений додаток, де вона агрегирується і передається на набір розподілених смартфонів.

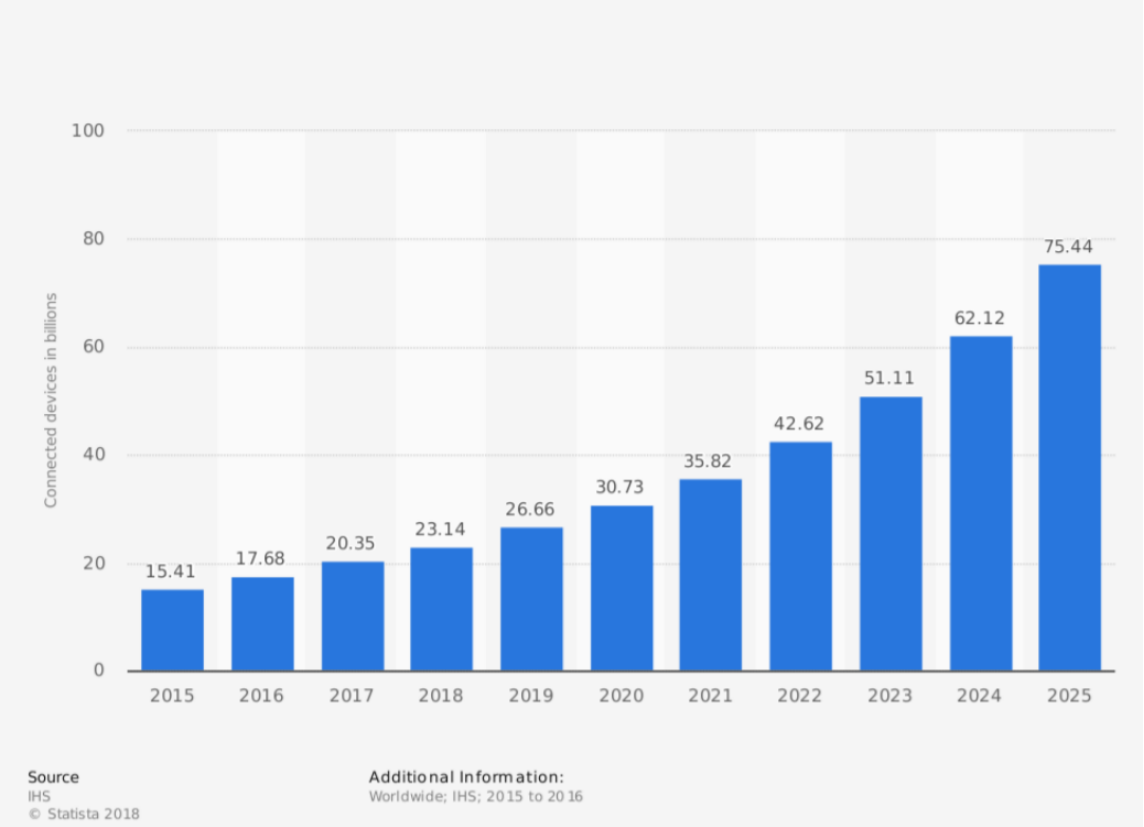

Рисунок 1.1 – Statista: Прогнозований ріст онлайн-пристроїв IoT до 2025 року. Оригінальне джерело: Statista Gmbh [9].

У той час, коли зв'язок між пристроями та процесами поліпшив і оптимізував виробничі процеси, що дозволило створити безліч нових додатків, темпи цього перетворення призвели до того, що безпеці цих систем приділяється надто мало уваги.

Ризики, які стосуються як безпеки, так і з конфіденційності, пов'язані з поширенням Інтернету в повсякденному житті, є тим, на що більшість населення в основному не звертає уваги. Insecam, вебсайт, який транслює онлайн-трансляції з тисяч IP-камер з обліковими даними за замовчуванням без відома власників, ілюструє серйозність цієї проблеми [12]. Пристрої та системи, які ніколи раніше не мали комунікаційних можливостей, підключаються до мережі, наражаючи її на значне потенційне втручання та зловживання таким чином, що це може вплинути на добробут всього населення. Найважливіші системи, від яких щодня залежить суспільство, такі як електричні мережі, медичні прилади та системи управління рухом, все частіше потрапляють в цю категорію, піддаючись атакам так, як ніколи раніше.

Деякі з найбільших проблем з безпекою цих пристроїв включають наступне:

1) Слабкі облікові дані за замовчуванням

Сьогодні основним фактором успіху більшості атак на пристрої IoT є той факт, що ці пристрої постачаються зі слабкими обліковими даними за замовчуванням, які ідентичні у всіх зразках даної моделі. Крім того, ці облікові дані зазвичай є єдиною мірою безпеки, яка використовується для захисту від загроз для цих пристроїв, тому зловмиснику потрібно досить мало зусиль, аби виконати успішний злам.

2) Обмеження в застосуванні оновлень

Середовища, в яких розгорнуто багато пристроїв IoT, дуже ускладнюють виправлення вразливостей і застосування оновлень до цих пристроїв. Як правило, велика кількість різнорідних пристроїв є географічно розкиданими, розподіленими по налаштуванню, і обмінюються даними по різних мережевих

протоколах. Застосування оновлень в такому розгортанні є дуже складним завданням.

3) Обмежена місткість для шифрування

Через обмежені обчислювальну потужність та пропускну здатність на багатьох IoT-пристроях з низьким енергоспоживанням розгортання шифрування на цих пристроях є практично неможливим. Інтенсивні обчислення, необхідні більшістю протоколів шифрування як для шифрування, так і для дешифрування передаваних даних, що зберігається, практично ускладнюють реалізацію шифрування. Щоб проілюструвати це, відкрите Міжнародне співтовариство проектувальників (IETF) у своїй новітній специфікації для глобальних мереж з низьким енергоспоживанням (LPWAN) заявила, що реалізація механізмів аутентифікації ключів стає «складною для обробки в LPWAN з обмеженою пропускною спроможністю» [13]. Як обговорювалось Даулінгом і ін., - «Якщо шифрування не реалізоване правильно, це може відкрити двері для атак на конфіденційність, аутентифікацію джерела і цілісність даних» [14].

4) Ненадійні протоколи зв'язку

Багато пристроїв IoT використовують ненадійні протоколи зв'язку через їх потреби в низькому енергоспоживанні. Мережевий протокол telnet, що працює через порт 23 / TCP, значною мірою застарів, оскільки він, передаючи весь трафік у вигляді відкритого тексту, включаючи облікові дані для входу, є досить ненадійним. Однак це простий протокол для розгортання на малопотужних пристроях, що означає, що він широко використовується в IoT-пристроях для аутентифікації. Це ще більше послаблює безпеку цих пристроїв: пасивний зловмисник, який підслуховував мережевий трафік, може перехопити дані пристроїв IoT для подальшого аналізу або поставити під загрозу сам пристрій.

Для зловмисників пристрої IoT дуже привабливі: вони завжди підключені до мережі, використовують слабкі механізми безпеки та часто мають доступ до потужних оболонок. Поява вільно доступних онлайн інструментів сканування,

таких як Shodan, означає, що ідентифікація цих пристроїв стає банальною. Як підтвердив провідний експерт з кібербезпеки Брюс Шнайер у своїй статті «Інтернет речей абсолютно невпевнений – і часто недоступний для виправлення у 2014 році», - «якщо ми не розв'яжемо цю проблему найближчим часом, нас чекає катастрофа в області безпеки, оскільки хакери розуміють, що легше зламати маршрутизатори, ніж комп'ютери» [17].

Хоча багато IoT-пристроїв являють собою пристрої з низьким енергоспоживанням, які зазвичай розгортаються в віддалених районах з низькою пропускною здатністю мережі, велика кількість цих пристроїв може надавати комбінований вплив, який дає значний ефект: введення ботнетів IoT. Ботнети IoT, ймовірно, є найбільш активним і актуальним типом ботнетів в даний час. Це ботнети, що складаються виключно зі скомпрометованих пристроїв IoT. Було встановлено, що більшість цих пристроїв використовують протоколи аутентифікації Secure SHell (SSH) і telnet з простими комбінаціями імені користувача і пароля за замовчуванням, що спрощує запуск атак аутентифікації методом грубої сили для отримання root-доступу до пристрою.

Наймасштабніша атака на сьогодні з боку ботнету IoT застала світ зненацька в кінці 2016 року. Ботнет Mirai, що зібрав армію з більш ніж 300000 пристроїв IoT користуючись їх погано реалізованими механізмами аутентифікації, провів один з найбільших у світі DDoS-атак. Атаки почалися з націлювання на Krebs on Security [19], веб-сайт відомого журналіста-розслідувача Брайана Кребса, зі швидкістю майже 1 Тбіт / с при атаці на французького провайдера хмарного хостингу OVH. Дослідження, проведені С. Антонакакіс та ін. "Про успіх Mirai" [20] виявив, що 3 найпоширеніших архітектур, в яких встановлено використання протоколу telnet, «камери безпеки, відеореєстратори та споживчі маршрутизатори представляють більшість» цих пристроїв.

Масштаб і вплив цього ботнету були революційними. Той факт, що використовуючи величезну кількість пристроїв з низькими обчислювальними можливостями, ботнет був здатний загрожувати найбільш захищеним цілям у

світі, говорить про масштаби проблем, з якими в даний час стикаються пристрої IoT. Як заявив Брюс Шнайер у своїй статті «Ботнет речей» 2017 року, «ботнети стануть більші і потужніші просто тому, що кількість вразливих пристроїв зросте на порядок протягом наступних кількох років ... в цілому, тенденції сприяють атакуючому» [21].

З часу атаки Mirai сплив ряд інших бот-мереж IoT, багато з яких являли собою варіанти вихідного коду Mirai, який був опублікований онлайн користувачем Anna Senpai на популярному сайті HackForums [22]. Ще один вартий уваги ботнет – BrickerBot, ботнет IoT, який з'явився незабаром після початкової атаки Mirai на KrebsOnSecurity [23]. Автор цього ботнету стверджує, що є хактівістом спільноти білих капелюхів, учасники якої прагнуть виконати те, що вони назвали «Інтернет-хіміотерапією»: повне знищення пристроїв IoT з погано реалізованими заходами безпеки як покарання для їх власників. У прощальному електронному листі, наданому сайту безпеки BleepingComputer від 10 грудня 2017 року, він попереджує, що світ повинен «прокинутися від того факту, що Інтернет знаходиться на одному чи двох кроках від того, щоб розчарувати всіх», а також про те, що організації і люди повинні вжити заходів, щоб запобігти цьому [24].

На рисунку 1.2 показаний фрагмент з дослідження «Understanding the Mirai Botnet», який ілюструє паролі, які найчастіше зустрічаються на всіх пристроях, проаналізованих у дослідженні.

Password	Device Type	Password	Device Type	Password	Device Type
123456	ACTi IP Camera	klv1234	HiSilicon IP Camera	1111	Xerox Printer
anko	ANKO Products DVR	jvbzd	HiSilicon IP Camera	Zte521	ZTE Router
pass	Axis IP Camera	admin	IPX-DDK Network Camera	1234	Unknown
888888	Dahua DVR	system	IQinVision Cameras	12345	Unknown
666666	Dahua DVR	meinsm	Mobotix Network Camera	admin1234	Unknown
vizxv	Dahua IP Camera	54321	Packet8 VOIP Phone	default	Unknown
7ujMko0vizxv	Dahua IP Camera	00000000	Panasonic Printer	fucker	Unknown
7ujMko0admin	Dahua IP Camera	realtek	RealTek Routers	guest	Unknown
666666	Dahua IP Camera	1111111	Samsung IP Camera	password	Unknown
dreambox	Dreambox TV Receiver	xmhdipc	Shenzhen Anran Camera	root	Unknown
juantech	Guangzhou Juan Optical	smcadmin	SMC Routers	service	Unknown
xc3511	H.264 Chinese DVR	ikwb	Toshiba Network Camera	support	Unknown
OxhlwSG8	HiSilicon IP Camera	ubnt	Ubiquiti AirOS Router	tech	Unknown
cat1029	HiSilicon IP Camera	supervisor	VideoIQ	user	Unknown
hi3518	HiSilicon IP Camera	<none>	Vivotek IP Camera	zlxx.	Unknown
klv123	HiSilicon IP Camera				

Рисунок 1.2 – Ботнет Mirai: найпоширеніші паролі за замовчуванням.

Оригінальне джерело: Understanding the Mirai Botnet [20].

1.1.3 Безпека об’єктів критичних інфраструктур

У той час, коли між багатьма могутніми державами світу посилюється політична напруженість, [25] кібератаки відіграють помітну роль у війнах і тероризмі. Кібератаки на критично важливі інфраструктури можуть привести до серйозних наслідків. Показуючи серйозність загроз з боку національних держав, які кункурують, Б. Унал та ін. [26] обговорили наростаючу залежність ядерної зброї від цифрових технологій та систем зв'язку, заявивши, що «вірогідність спроб кібератак на системи ядерної зброї... зростає в результаті постійних погроз з боку держав і недержавних груп».

В опублікованому у 2017 році звіті, що містить рекомендації для адміністрації США з національної кібербезпеки, дослідник Джоел Бреннер стверджує [27], що одне з восьми основних завдань, які стоять перед урядом Сполучених Штатів в даний час, полягає в тому, щоб «дозволити операторам критично важливої інфраструктури швидко виявляти та реагувати на кібер-ризики, що виникають через міжсекторні зв’язки, а також з їх власної мережі». Це твердження точно зображає проблеми, з якими сьогодні стикаються постачальники критично важливих послуг у всьому світі, і це відбивається в інцидентах по всьому світу з ростом числа атак на високопоставлені цілі.

Зараз є зрозумілішим, ніж будь-коли, що ключовим організаціям і критично важливим службам необхідно знову зосередитися на стратегіях захисту своїх IT-інфраструктур. Однією з таких категорій критичних інфраструктур є системи диспетчерського управління та збору даних (SCADA), які використовуються в промислових середовищах управління для виконання таких функцій, як управління процесом і моніторинг. Як правило, ці системи працюють з безліччю компонентів, включаючи датчики і механічні деталі, такі як двигуни, що підвищує ефективність роботи в цих умовах.

Забезпечення безпеки цих систем життєво необхідно: вони є основною системою управління практично усіма критично важливими для держави інфраструктурами, такими як транспорт, електроенергія і водопостачання.

Ясно, що наслідки компрометації цих систем можуть бути серйозними та далекосяжними.

У травні 2017 року в Великобританії відбулася одна з найбільш пам'ятних атак в останні роки на службу, критичну до держави. Глобальні атаки WannaCry Ransomware, які успішно скомпрометували понад 200 000 систем у більш ніж 100 держав, пошкодили системи в Національній службі охорони здоров'я Великобританії (NHS) на 37 сайтах в автономному режимі більш ніж на тиждень, і за цей час було скасовано понад 6 912 призначень. Хоча NHS майже напевно не була конкретною ціллю авторів, атаки справили величезний вплив на служби охорони здоров'я по всій Великобританії [29].

На рисунку 1.3 зображена ілюстрація зараження систем на сайтах лікарень NHS вимагачами WannaCry. Інфекція не обмежувалася користувацькими пристроями та серверами, зачіпаючи такі системи, як сканери МРТ і пристрої для аналізу крові [30].

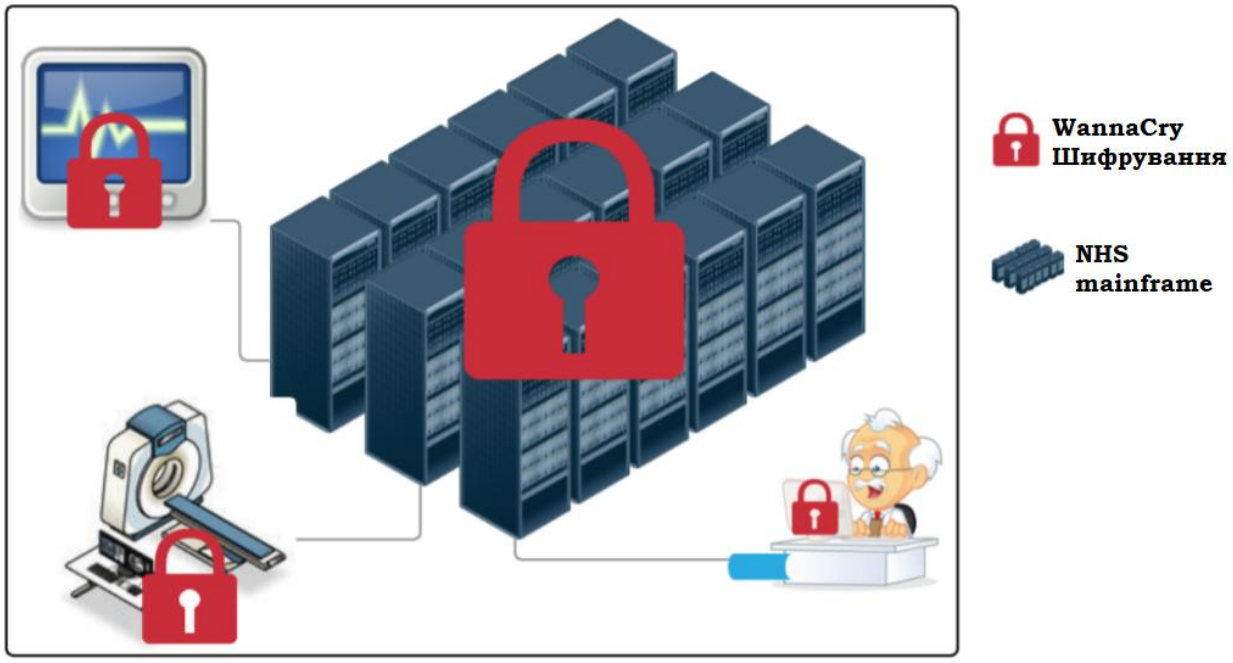

Рисунок 1.3 – Ілюстрація інфекції WannaCry в NHS

Дослідження компрометації інфраструктури NHS за допомогою вимагачів WannaCry являє собою міцну основу для розв’язання багатьох проблем, що

виникають в критично важливих інфраструктурах обслуговування. Бувши настільки значним інцидентом, він підкреслює серйозні наслідки, які кібератака може мати для критично важливої служби. У звіті про аудит, опублікованому Національним аудиторським управлінням Великобританії після інциденту [29], відзначений ряд істотних недоліків в стратегіях захисту безпеки ДСЗ до 12 травня 2017 року:

1) Не було ніякого моніторингу інцидентів на місці, щоб попередити осіб, що відповідають за безпеку, про який-небудь незвичну поведінку в системах NHS: це випливає з того факту, що було потрібно більше півдня, щоб дістатися до тих, хто міг вжити заходів щодо виправлення положення.

2) Не було ніякої централізації системних даних, що означало, що експерти з безпеки повинні були фізично відвідувати порушені сайти, щоб зібрати важливі дані про атаки. Це ще більше продовжило період, протягом якого ці системи залишалися непрацездатними.

3) Рекомендації щодо оновлень для IT-систем NHS були випущені до інциденту, однак не виконувалися. На додаток до цього не було реалізовано ніякої системи, щоб це визначити. Застарілі та погано налаштовані операційні системи були основним фактором успіху компрометації, повністю заснованої на експлойтах Windows.

У загальному підсумку, відсутність моніторингу в системах NHS означала, що коли відбувалися атаки, системи на заражених сайтах залишалися в автономному режимі протягом тривалого періоду часу, викликаючи хаос у всій системі. Ясно, що для захисту як своїх власних систем, так і систем, які залежать від них, організації повинні приділяти особливу увагу навчанню безпеці своїх членів, безпеці та обслуговуванню своїх технологій і строгості управління своїми системами.

1.2 Системи виявлення вторгнень

Система виявлення вторгнень (IDS) – це додаток безпеки, який використовується для виявлення спроб зловмисника отримати несанкціонований доступ до системи або системного ресурсу. Він зазвичай розгортаються в регіоні, відомому як демілітаризована зона (DMZ), тобто підмережі, що відокремлює внутрішню мережу від ненадійних зовнішніх мереж, забезпечуючи додатковий рівень ізоляції між внутрішньою і зовнішньою системами. Більшість традиційних IDS працюють на основі порівняння дій з певною політикою безпеки, а також дозволяють або забороняють дію на цій основі.

IDS відіграють важливу роль в захисті як корпоративних, так і особистих систем. Серед багатьох переваг наступні:

- Виявивши вторгнення на ранніх стадіях, порушник може бути видалений з системи до того, як буде завдано якоїсь шкоди.
- Зловмисники в цілому прагнуть залишатися анонімними та непоміченими. Якщо потенційний зловмисник знає, що в системі є ефективна IDS, це може утримати його від атаки.
- Може бути зібрана цінна інформація про характер вторгнень й про те, як вони відбулися, що дозволяє адміністраторам постійно переглядати свої системи і стратегії безпеки.

Те, як надається кожна з цих переваг і в якій мірі, залежить від типу використовуваних IDS. Слід визнати, що загрози можуть виходити від вузлів, як внутрішніх, так і зовнішніх стосовно даної мережі, і тому максимально ефективний IDS буде враховувати обидва напрямки.

IDS можуть бути широко класифіковані як пасивні або активні:

- Пасивні IDS не роблять активних дій щодо запобігання або втручанню в дії зловмисника.
- Активні IDS активно взаємодіють з атакуючим, щоб протистояти атаці.

У цьому дослідженні детально обговорюються два конкретних IDS: міжмережеві екрани (брандмауери) і honeypots.

1.2.1 Міжмережеві екрани

Міжмережеві екрани є найбільш поширеним типом IDS і використовуються практично в усіх корпоративних мережах як частина забезпечення безпеки мережі організації. В цілому вони класифікуються як пасивний механізм захисту мережі. Згідно з літературним оглядом Воронкова та ін. «Брандмауер – це система, що складається з програмного і / або апаратного забезпечення, яка розроблена для запобігання несанкціонованого доступу до пристрою або мережі/з неї» [31].

Основний принцип роботи брандмауера полягає в тому, щоб діяти як бар'єр на периферії мережі, запобігаючи доступу зловмисників до системи, яка захищається. На практиці вони фільтрують мережеві пакети на основі певної політики безпеки, яка являє собою набір налаштованих правил, і або приймають, або відхиляють даний пакет відповідно до цієї політики.

Ілюстрація типової корпоративної мережі, в якій на внутрішніх і зовнішніх точках входу в демілітаризовану зону (DMZ) використовуються брандмауери, зображена на рисунку 1.4:

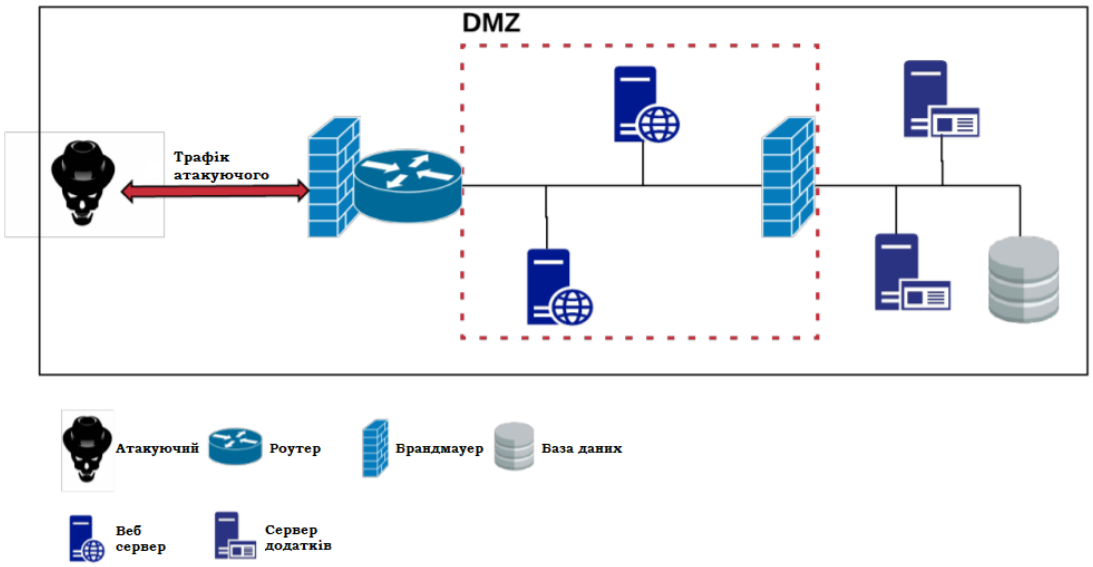

Рисунок 1.4 – Використання брандмауерів в демілітаризованій зоні

Ряд загальноприйнятих підходів до реалізації політик брандмауера визначено Омаром Сантосом в статті «Комплексний in-depth захист мережі» [32]. Деякі з них описані в такий спосіб:

- Шаблон відповідності

Брандмауер шукає фіксовану послідовність байтів в пакеті, часто вирівняну в певній позиції, яка відповідає полю в пакеті. Потім він приймає рішення про фільтрацію, звертаючись до своєї політики безпеки.

Одним з основних недоліків зіставлення зі зразком є те, що воно зазвичай демонструє високий рівень помилкових спрацьовувань, що робить його відносно неточним як метод класифікації пакетів, доступ до яких повинен бути заборонений.

- Аналіз протоколу

Аналіз протоколу виконується шляхом декодування специфічного для протоколу зв'язку, тобто шляхом перевірки пакетів, які відповідають конкретним протоколам зв'язку. Брандмауер ідентифікує елементи протоколу і перевіряє їх на предмет порушення, наприклад, шляхом перевірки явних полів в пакетах. Прикладом для SMTP-пакету може бути перевірка таких полів, як *HELO, MAIL, RCPT, DATA, SET, NOOP* і *QUIT*.

- Евристичний аналіз

Цей метод, також відомий як метод виявлення зловживань, намагається ідентифікувати зловмисників на основі набору відомих небажаних поведінок і шаблонів, які можуть бути визначені відповідно до правил. Системи, побудовані на цій концепції, серйозно обмежені в можливостях виявлення вторгнень, оскільки вони не можуть виявити вторгнення, для якого не визначено жодне правило.

- Аналіз на основі аномалій

Методи аналізу на основі аномалій намагаються визначити поведінку звичайного користувача з плином часу і порівняти поведінку всіх користувачів з ним, щоб виявити підозрілу поведінку, яка може бути пов'язане з вторгненням.

- Глибока перевірка пакетів

Deep-Packet Inspection (DPI) охоплює ретельне вивчення інформації, вбудованої в мережеві пакети, щоб приймати більш обґрунтовані рішення про те, як обробляти пакет. Це часто дозволяє більш ефективно протидіяти атакам, таким як DDoS, де DPI дозволяє більш точно ідентифікувати пакети, які мають небажаний вміст або в деякому роді не є правильними.

Для виявлення вторгнень лише з використанням брандмауерів існує ряд проблем. Хоча в якийсь момент ці системи точно відбивали конфігурацію корпоративних мереж, в сучасному світі Інтернету багато з передумов використання брандмауерів більше просто не мають місця. Як зазначено Стівеном М. Белловін в «Мислення безпеки: Зупинити хакерів наступного року» [33], розробка та використання брандмауерів засновані на таких передумовах:

1) Брандмауер працює в топологічної дросельної точки в мережі, що захищається, так що він розділяє два розділи мережі.

2) Усі вузли, що захищаються всередині брандмауера, мають однакову політику безпеки.

3) Всі вузли усередині брандмауера є довіреними.

Якщо всі ці умови виконуються, то брандмауер в системі буде працювати добре. Однак, приймаючи сьогодні будь-яку сучасну IT-інфраструктуру в організації, більшість з цих припущень, якщо вони взагалі є, не відповідають дійсності.

Крім того, брандмауерам стає все складніше використовувати методи перевірки пакетів для фільтрації мережевого трафіку через несумісність з іншими механізмами безпеки. Підвищене використання шифрування TLS і прагнення до шифрування заголовка пакета викликає величезні проблеми для роботи міжмережевих екранів, які не можуть розшифрувати пакети, щоб прийняти необхідні рішення по фільтрації, від яких це залежить.

Немає сумнівів в тому, що використання брандмауерів вже не є достатнім заходом для забезпечення безпеки сучасних мереж. Вони можуть бути досить

ефективними проти відомих, менш витончених атак і менш ефективними проти більш витончених цільових атак, які з більшою ймовірністю використовують нові експлойти. Як пояснює Фред Шнайдер у статті «План науки про кібербезпеку», «захищена система повинна захищатися від усіх можливих атак, в тому числі невідомих захиснику. Але захисники, володіючи обмеженими ресурсами, зазвичай розробляють захист тільки для атак, про які вони знають» [34].

В цілому, брандмауери найбільш ефективні при використанні в поєднанні з іншими механізмами безпеки, але їх нездатність класифікувати нові загрози та пом'якшувати їх наслідки є основним обмеженням щодо їх використання [33].

1.2.2 Приманки (Honeypots)

Відомий експерт з безпеки та засновник Honeynet Project Ленс Шпіцнер визначає Honeypot як «ресурс безпеки, цінність якого полягає в тому, щоб його досліджували, атакували або зламували». Для боротьби зі зловмисниками Honeypots використовують концепцію обману. Як пояснюється в документі Кліффа та ін., «Основні частини кібер-обману містять в собі спеціально створену захисником інформацію (яка буде використана для введення в оману) і неправильні дії, вжиті противником в результаті обману» [35].

Простіше кажучи, Honeypots – це пристрої, які маскуються як законні, і системи, які використовуються для виявлення, відстеження та аналізу моделей поведінки користувачів при несанкціонованому доступі до системи. Вони відносяться до категорії активних механізмів захисту мережі та розгорнуті виключно з метою бути атакованими. Ключовою характеристикою успішного honeypot є привабливість для зловмисника. «Привабливість» в цьому контексті означає, що приманка повинна виглядати як пристрій, який шукає зловмисник, тобто такий, що може легко експлуатуватися, пропонуючи при цьому максимальну цінність для зловмисника.

Вважається, що Honeypots діють як приманки та датчики в мережі, в якій вони розгорнуті.

Основні переваги використання технологій honeypot:

1) Здатність до конфігурації (Гнучкість до налаштування)

В цілому, Honeypots легко налаштовуються і можуть бути створені для імітації реальних систем. Ця простота в конфігурації дозволяє тим, хто їх розгортає, адаптувати свою стратегію захисту для вирішення атак, що виникають.

2) Розміщення всередині мережі

Honeypots розгортаються всередині інфраструктури системи, яку вони захищають, а не на периферії, як у випадку з брандмауерами. Це дозволяє забезпечити безпеку набагато ближче до реальних систем, на які націлені зловмисники.

3) Можливості ведення журналу

Honeypots дають унікальну можливість збирати цінну інформацію про характер атак, які скоєно проти системи, забезпечуючи можливість слідкувати за атакуючими без виявлення самого факту спостереження. Це дає системним адміністраторам більше шансів бути проінформованими щодо мінливих вимог безпеки своїх систем.

4) Мінімальна кількість помилкових спрацьовувань

Передумова використання Honeypots полягає в тому, що з ними ніхто не повинен взаємодіяти: немає ніяких легальних підстав для взаємодії, оскільки вони розгортаються з єдиною метою – залучення атак. Це означає, що, як правило, Honeypots викликають дуже мало помилкових спрацьовувань, оскільки кожна взаємодія автоматично визначається недовірливою. Це ще одна додаткова перевага для системних адміністраторів, оскільки зникає потреба фільтрувати великі файли журналів для ідентифікації значних подій, що є звичайною практикою для брандмауерів і багатьох інших IDS.

Однак, важливо враховувати, що люди всередині організації можуть помилково взаємодіяти з приманкою з цікавості, що часто може статись, коли

переглядаються захоплені записи. З цієї причини, наявність приманок в системі не повинна бути добре відомою.

Конструкція Honeypots, відповідно до контексту розгортання, є ключем до їх здатності забезпечити ефективне виявлення вторгнень в систему. Основні категорії конструкції приманок були викладені в статтях І. Мокубе та ін. [36].

Рівні інтерактивності приманок є важливим фактором, який визначає здатність атакуючого з ними взаємодіяти й, відповідно, обсяг та тип інформації, яка може бути зібрана. Рівні взаємодії варіюються від простого встановлення з'єднання до можливості завантаження і встановлення шкідливого ПЗ. Співвідношення вартості та ефективності приманки збільшується пропорційно, що означає, що високо інтерактивні Honeypots, ймовірно, буду дорогими для розміщення та обслуговування.

Будь-які ефективні приманки мають бути здатними взаємодіяти на якомусь рівні з атакуючим, а також спокійно спостерігати за його діями. Існує три практичних класифікаційних рівнів інтерактивності Honeypots: низький, високий і середній.

- Низький

Honeypots з низьким рівнем взаємодії найчастіше використовуються для попередження про те, що атака відбулась. Вони не надають ніяких засобів взаємодії зі зловмисником або збору даних про атаку і використовуються в тих випадках, коли переважним є рішення з більш низьким рівнем ризику. В загальному, приманки з низьким рівнем взаємодії просто емулюють реальний сервіс, і тому не дають ніякої можливості для злому системи зловмисником. Це може бути основним обмеженням їх використання, оскільки від запобігання подальшої взаємодії зі зловмисником можна отримати дуже мало знань.

- Високий

Honeypots з високим рівнем взаємодії – це повноцінні системи, які дають зловмисникам можливість використовувати реальні програми та сервіси. Їх використання дозволяє отримати повну і детальну інформацію про природу атак. Однак важливим недоліком є високий ризик, пов'язаний з тим, що

зловмисник атакує реальний пристрій, який може бути використаним для подальших атак на інші пристрої.

- Середній

Honeypots із середнім рівнем взаємодії пропонують гарне проміжне положення, використовуючи емуляцію компонентів реальної системи, щоб забезпечити деякий рівень інтерактивності з атакуючим, при цьому не піддаючи небезпеці реальні системи, які можуть бути скомпрометовані.

Таблиця 1.1 – Короткий опис кожного окремого рівня інтерактивності приманок

Рівень інтерактивності	**Якість Отриманої інформації**	**Ризик**	**Вартість**
Високий	Відмінна	Високий	Висока
Низький	Низька	Низький	Низька
Середній	Хороша	Середній	Відносно низька

Існує два сценарії розгортання приманок: виробничий і досліджувальний. Головним чином, вони розрізняють функцію, яку приманка повинна забезпечити для системи, в якій вона буде розгорнута.

- Виробничі приманки

Як правило, виробничі приманки розгортаються в великих корпоративних мережах з метою зробити їх частиною активного захисту мережі в інфраструктурі організації. Приклад такого розгортання показаний на рисунку 1.5.

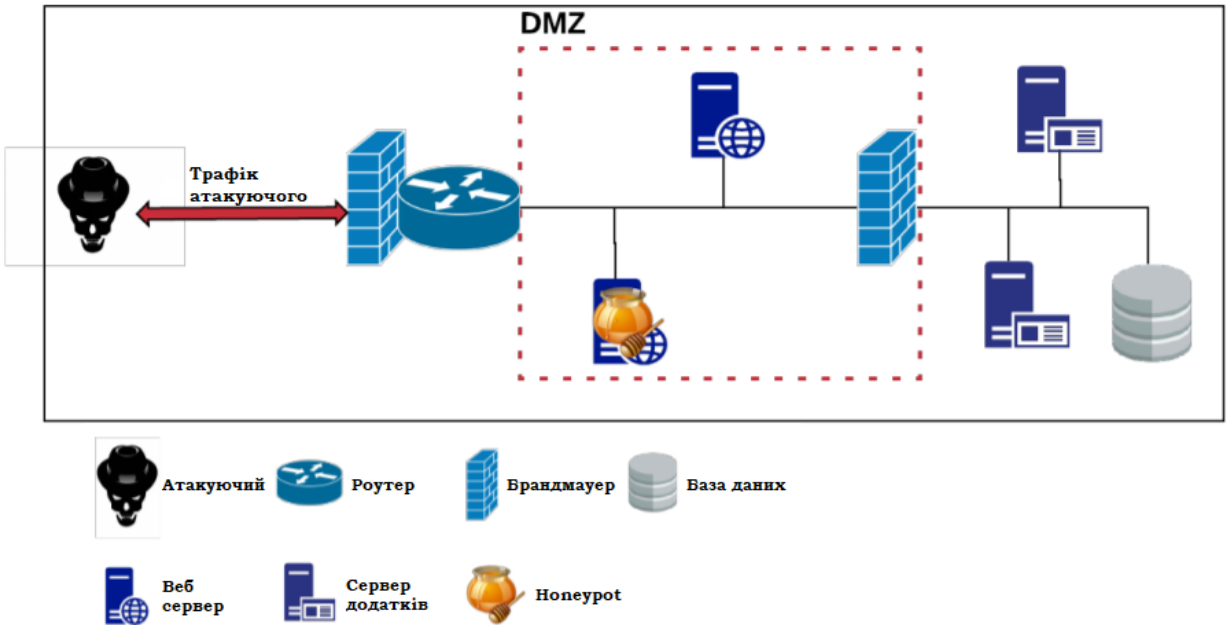

Рисунок 1.5 – Використання Honeypots у демілітаризованій зоні

Основна мета такого розгортання – діяти як приманка, відволікаючи зловмисника від цінних машин в мережі за допомогою, здавалося б, більш цінної і вразливої цілі. Це дозволяє приманці завчасно попереджати системних адміністраторів про факт вторгнення, що дає їм можливість ізолювати цінні пристрої в мережі. Як зазначалося раніше, таке розгортання також дозволяє системним адміністраторам визначати вразливі точки в їх інфраструктурі.

Ідея залучення зловмисників в систему, спонукання їх взаємодіяти з нею і демонстрації стратегії атак, не завдаючи ніякої шкоди системам з реальною цінністю, є головною привабливістю використання приманок у виробничому середовищі.

- Дослідницькі приманки

Як випливає з назви, ці приманки, як правило, використовуються в цілях дослідження, а не безпеки. Вони акцентуються не стільки на здатності діяти як приманка, скільки на здатності збирати цінні дані для аналізу. Дозволяючи зловмисникам взаємодіяти та інфікувати Honeypots, можна проводити дослідження, що стосуються поведінки й стратегій зловмисників.

У порівнянні з традиційними пасивними механізмами виявлення вторгнень, такими як брандмауери, Honeypots демонструють деякі суттєві відмінності:

- Брандмауери пасивно визначають всіх зловмисників і просто попереджають про те, що стався інцидент безпеки (наприклад, спроба несанкціонованого підключення). Системний адміністратор потім може, наприклад, помістити вихідний IP-адрес з'єднання в чорний список, щоб заборонити йому подальший доступ до системи. Однак, якщо виникає нова атака, для якої немає правил, визначених у політиці безпеки, брандмауер впоратися з нею не зможе. Це ілюструє, як брандмауери забезпечують протидію лише відомим загрозам.
- На відміну від брандмауерів, приманки активно спонукають зловмисників видавати свої стратегії та наміри атак, фіксуючи усі їх дії. Крім того, Honeypots здатні залучатись для боротьби з раніше невідомими загрозами.

Як зазначив С. Белловін, «світ змінився... рішення, на яке можна було покластися раніше (брандмауери), слід переглянути й, можливо, відмовитися» [33]. Попри це, брандмауери не застаріли та будуть продовжувати відігравати важливу роль у безпеці ІТ й мереж завдяки своїй здатності виявляти і класифікувати потенційні ризики безпеки на границі мережі, яка ними захищена. Однак найкраще використовувати їх в поєднанні з активними механізмами захисту мережі.

Мережа взаємопов'язаних приманок, яка координує свої зусилля для забезпечення активного захисту мережі, називаються Honeynet. З огляду на те, що мережа складається з декількох приманок, у неї є додаткові можливості для збору цінної інформації про атаки щодо загроз. Наприклад, вона дозволяє вивчати поширення атак від однієї приманки до іншої, моделювати варіанти різних систем, які можуть залучати різні категорії атак тощо.

Важливим елементом мережі Honeynet, який описаний Л. Шпіцнером в статті «Honeypots: Спіймати інсайдерську загрозу» [38] є так званий «медовий

шлюз» (далі – Honeywall) – мостовий пристрій між зовнішніми системами і, власне, Honeynet. Доступ до Honeynet можливий лише через Honeywall, який може відстежувати і контролювати мережевий трафік в і з приманок в мережі, що дозволяє виконувати додатковий моніторинг дій зловмисників.

Схема, взята зі статті Л. Шпіцнера, в якій пропонується архітектура з використанням Honeynet, зображена на рисунку 1.6:

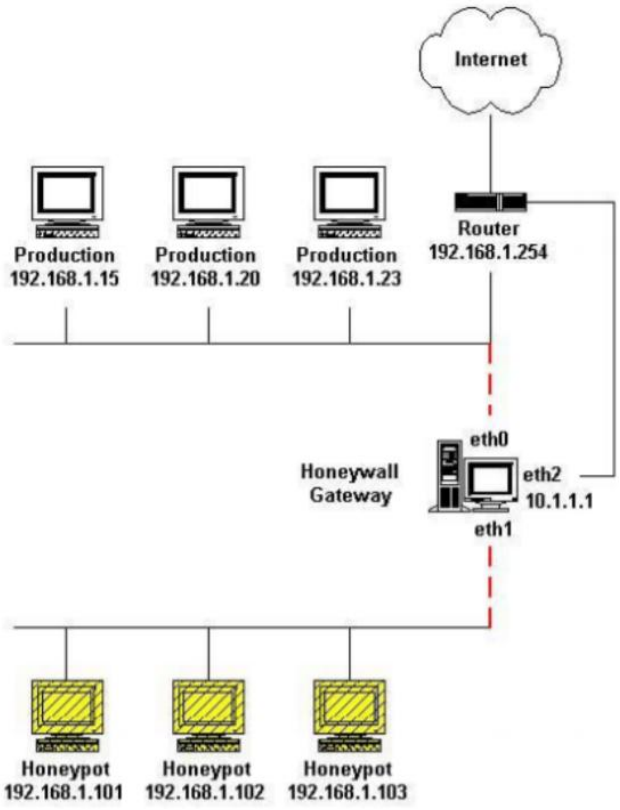

Рисунок 1.6 – Запропонована Лансем Шпіцнером архітектура Honeynet.
Оригінальне джерело: Honeypots: Catching the Insider Threat [38].

Таким чином, використання Honeynet дозволяє розгорнути високо контрольовану мережу активного захисту, в якій усі дії зловмисників стають видимими. Результати, які можна отримати, спостерігаючи за поведінкою зловмисників в необмеженому мережевому середовищі, є дуже цінними для тих, хто хоче постійно покращувати захист своїх систем. Однак ризик, пов'язаний з розміщенням системи такого типу, є основним недоліком, оскільки у мережі навмисно вразливих пристроїв існує набагато більше точок відмови.

З широким розгортанням приманок в мережах існує ряд проблем:

1) Проблема виявлення

Однією з основних проблем розгортання приманок є ризик виявлення системи як несправжньої. Це особливо актуально для приманок з низьким і середнім рівнем взаємодії, адже точно імітувати поведінку реальної системи неможливо. Це означає, що ретельно сплановані дії зловмисника можуть дозволити виявити, що він взаємодіє не з реальною системою.

Однак, Honeynet піддається цьому ризику менше, ніж окремі приманки, оскільки він надає мережу взаємопов'язаних приманок з високим рівнем взаємодії з реальними сервісами. Така система дуже приваблива для зловмисників через розширені можливості, які вона надає, і, таким чином, менш імовірно, що вона не є справжньою системою.

2) Проблема прийняття

Ідея запросити зловмисника в виробничу систему здається потенційно руйнівною, тому багато системних адміністраторів просто не розглядають Honeypots як частину своєї інфраструктури безпеки. Canary, рішення для приманок, розроблене Thinkst Applied Research, підкреслює на цільовій сторінці свого продукту проблему впровадження Honeypot у великих мережах: «При всіх існуючих проблемах в мережі нікому не потрібна ще одна машина для адміністрування та занепокоєння» [39]. Хоча можна вважати, що дане твердження є комерційним кроком, в ньому підкреслюється потреба і недолік здійсненних рішень для великих організацій щодо розгортання та обслуговування систем на основі Honeypot.

3) Етичні проблеми

Використання будь-якої технології спостереження, яка відноситься до категорії, в яку потрапляють приманки, пов'язане з етичними питаннями. Honeypots широко поширені як етичний підхід до виявлення вторгнень і розуміння кібератак. Оскільки немає законних причин для взаємодії з приманкою, будь-які взаємодії, ймовірно, мають шкідливий характер. В цьому випадку Honeypots просто служить для протидії подальшим атакам такого типу

в майбутньому. Ці етичні питання детально обговорювались у роботах І. Мокубе та ін. [36]. Однак, з обговорення в офіційному документі, опублікованому Інститутом системного адміністрування, мереж і безпеки (SANS), можна зробити висновок, що їх використання не включає:

- Захоплення, оскільки немає стимулу атакувати приманку – те, до чого уразливі всі пристрої, тому що зловмисники завжди будуть прагнути атакувати пристрої в межах їх досяжності, незалежно від того, чи це приманка. Як влучно сформулював дослідник безпеки Л. Шпіцнер [41], «зловмисники знаходять і зламують приманки за власною ініціативою»
- Вторгнення в особисте життя, оскільки механізми спостереження в цілому вважаються прийнятними, якщо вони слугують просто для захисту свого середовища. У фізичних середовищах часто зустрічається приклад використання камер відеоспостереження.

В цілому, етичні проблеми, пов'язані з використанням приманок, загальновизнано вирішуються у зв'язку з характером розміщення приманок: в кінцевому підсумку їх мета полягає в захисті від неетичних дій зловмисників.

1.2.3 Моніторинг кіберінцидентів

Моніторинг кіберінцидентів є важливою частиною забезпечення ефективного виявлення вторгнень. Монітори кіберінцидентів – це платформи, які допомагають системним адміністраторам виявляти та реагувати на інформацію, зібрану за допомогою механізмів виявлення вторгнень.

Широко визнано, що здатність користувача легко використовувати системи безпеки вносить вирішальний вклад в їх ефективність. Як стверджують Сассе М. А. та ін., «Механізми безпеки часто занадто трудомісткі, щоб люди могли з ними працювати, або настільки складні, що навіть ті, хто хочуть їх використовувати, роблять помилки» [42]. Крім того, використання механізмів безпеки майже без винятку додає витрати.

В літературному огляді, присвяченому корисності та простоті використання брандмауерів, Воронков та ін. виявили, що дослідження в цьому напрямку можуть значно підвищити ефективність механізмів безпеки, оскільки їх налаштування – «це складний, схильний до помилок процес... неправильне налаштування брандмауерів призводить до широкого числа вразливостей в мережі; середовища з розподіленими та декількома міжмережевими екранами тільки погіршують ситуацію» [31].

Методи візуалізації стали критично важливими для надання корисних рішень безпеки. Проблеми безпеки складні за своєю природою: у випадку таких інструментів, як брандмауери, які працюють на основі списку певних правил, стає дуже важко розуміти та підтримувати загальну політику безпеки в міру росту списку.

Результатом використання візуалізації в механізмах безпеки є те, що відомості про загрози передаються чіткіше і ефективно тим, хто використовує дані. Простота видимості стану системи є найважливішим елементом активного захисту мережі при виявленні та розслідуванні потенційно шкідливих дій і виявленні невідомих загроз до того, як вони зможуть завдати шкоди. Як пояснив Е. Василоманолакіс та ін. у своєму дослідженні, яке детально описує розробку системи моніторингу інцидентів на основі Honeypots, така система «об'єднує різні генератори сповіщень в єдину систему, яка допомагає користувачеві приймати обґрунтовані рішення» [43].

1.3 Сучасна системна інфраструктура

Існують значні зміни в рішеннях для хостингу, які використовуються великими організаціями, які значною мірою покладаються на свою IT-інфраструктуру, особливо для тих, хто надає послуги на такій платформі. Аутсорсинг управління ресурсами в даний час є найкращим підходом для багатьох великих організацій, враховуючи, що спеціалізовані хостинг-

провайдери все частіше пропонують комплексні рішення для управління обладнанням, мережею та інфраструктурою безпеки.

Архітектура цих розподілених сервісів також розвивається. Концепція розділення сервісів шляхом їх розгортання у вигляді мікросервісів дозволила підвищити ефективність роботи організацій і відмовитись від схильних до збоїв монолітних конструкцій минулого.

1.3.1 Інфраструктура як послуга (IaaS)

У минулому організації розміщували свої системи на серверах і фізичній інфраструктурі, якою вони володіли і керували. Цей підхід був надто ресурсомістким, що і мотивувало розвиток Infrasracrure-as-a-Service (скорочено, IaaS). IaaS – це сервісна модель, в якій фізична інфраструктура, включаючи обладнання, сховище, сервери, простір центру обробки даних і мережеві компоненти, надається організаціям на аутсорсинг, а використання хмарних платформ, що дозволяють організації розміщувати свої сервіси на обладнанні іншої організації, пропонує для цього практичне рішення.

Переваги для організацій, що передають управління своєю інфраструктурою стороннім постачальникам, включають наступне:

- Існує гарантія максимального часу безвідмовної роботи, оскільки хмарні платформи є платним сервісом.
- Відсутня необхідність в попередніх капіталовкладеннях, оскільки немає ніяких витрат на установку або обслуговування, пов'язаних з оновленням або розширенням обладнання.
- Існує широкий спектр географічних місць розташування, доступних для розгортання, оскільки хостингові служби розташовані по всьому світу, що дозволяє організаціям розміщувати свої послуги ближче до географічно розподілених користувачів.

Хмарні платформи, як правило, використовують віртуалізацію базового апаратного забезпечення «з нуля» за допомогою використання віртуальних

машин (ВМ), які запускають програмне забезпечення поверх фізичних серверів для емуляції конкретної апаратної системи і є допоміжним компонентом IaaS. У будь-якій віртуалізованій системі існує ряд компонентів:

- Хост VM, який є сервером, що підтримує віртуалізацію;
- Гіпервізор або монітор віртуальної машини, який знаходиться між фізичним хостом і віртуальною машиною;
- Віртуальні машини, які є ізольованим програмним середовищам, що імітує виділені системи з «чистим залізом» (bare-metal systems).

Віртуальні машини відносно не прості в обслуговуванні і дорогі, оскільки кожна окрема одиниця підтримує свій власний образ операційної системи (ОС) – копію всього стану ОС, яка зберігається в постійній, незалежній формі.

Відсутність гнучкості також обмежує платформонезалежність: досить складно скопіювати на іншу віртуальну машину одну і ту ж конфігурацію середовища.

Концептуальна схема, яка показує компоненти віртуалізованого середовища, зображена на рисунку:

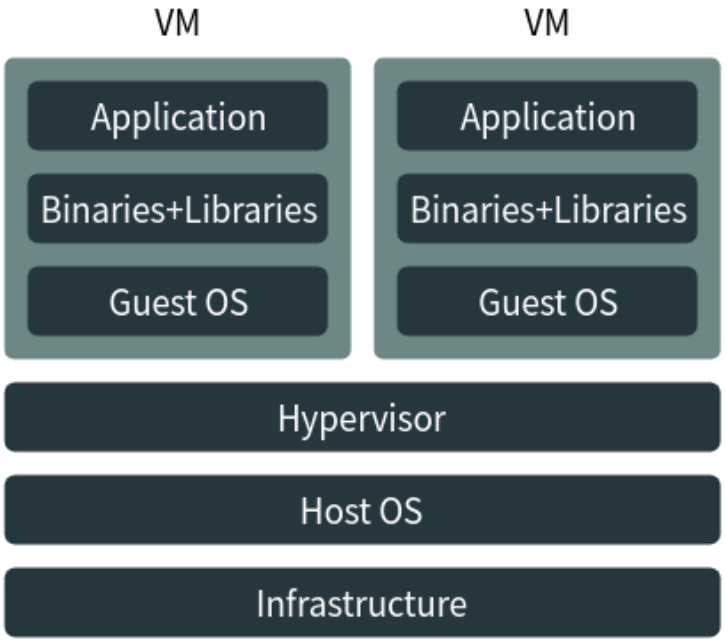

Рисунок 1.7 – Компоненти віртуалізованого середовища. Оригінальне джерело: Boolean World [44].

1.3.2 Платформа як послуга (Platform-as-a-Service)

Ідея PaaS полягає в тому, що користувачі можуть зосередитися на запуску і розробці додатків замість управління інфраструктурою, яка їх підтримує. В системах PaaS користувачам надається віртуальна інфраструктура, на якій вони можуть розгортати будь-який додаток, не турбуючись про те, як йому виділяються ресурси.

Контейнер – це ізольоване середовище виконання, визначене одним пакетом, який містить все необхідне для його запуску: код, час виконання, системні інструменти, системні бібліотеки, налаштування тощо. Це тип віртуалізації на рівні ОС, який ізолює додаток від інфраструктури хоста, на якому він виконується. Додаток і всі його залежності можуть бути упаковані в образ контейнера, який згодом буде опублікований в реєстрі контейнера, щоб необмежена кількість користувачів могла працювати в ідентичних середовищах додатка.

Концептуальна схема, яка показує компоненти контейнера, зображена на рисунку:

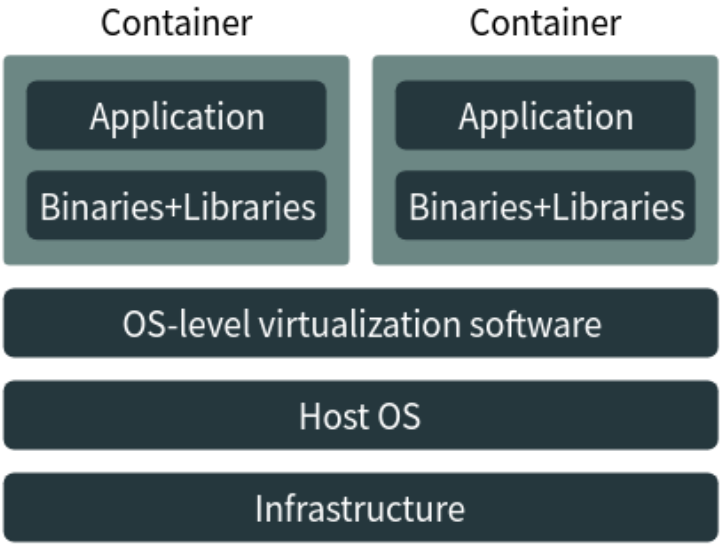

Рисунок 2.1 – Компоненти контейнера. Оригінальне джерело: Boolean World [44].

Серед безлічі переваг використання контейнерів в порівнянні з віртуальними машинами, можна виділити наступні:

1) Контейнерам не потрібно зберігати власну копію образу ОС, оскільки вони спільно використовують ядро операційної системи хоста.

2) Всі залежності, які необхідні додаткам, упаковані в один образ. Це означає, що платформонезалежність прикладних середовищ, яку так важко досягнути за допомогою віртуальних машин, вбудована в контейнери.

3) Контейнери постачаються з власним мережевим стеком і сховищем без додаткових витрат на створення і запуск віртуальної машини.

На сьогодні, досить багато організацій переносять розгортання своєї інфраструктури з віртуальних машин на контейнери. Недавнім прикладом цього є Netflix, який почав міграцію хостингу своєї інфраструктури в хмару AWS ще у 2008 році. На момент написання статті, у 2019 році Netflix переносив хостинг всіх своїх додатків з віртуальних машин на контейнери. У статті, в якій обговорювався цей зсув в інфраструктурному підході Netflix, А. Леунг та ін. стверджують [45], що «одна з переваг використання контейнерів... полягає в тому, що вони абстрагують велику частину машинно-орієнтованого управління, яке додатки виконували в віртуальних машинах». Це одна із помітних переваг використання контейнерів над віртуальними машинами: тим, хто розгортає додаток, більше не потрібно створенні та запуску додатків.

Область, в якій ще належить виконати велику роботу, полягає в тому, щоб впровадити програми для забезпечення безпеки в поступово зростаючу контейнерну екосистему. Існує величезний потенціал для контейнеризації додатків безпеки.

- Одноразовість

Контейнери є практично одноразовими: якщо контейнер інфікувався шкідливим ПЗ, його легко повністю видалити і повторно розгорнути з мінімальним порушенням роботи системи.

- Ізоляція

Контейнери використовують функціональність Linux, відому як «простір імен», розміщуючи додатки, які на них працюють, в обмежених середовищах, ізольованих одне від одного. Окремі контейнери запускаються у своєму власному незалежному просторі імен і отримують власне незалежне уявлення про різні ресурси операційної системи. Також контейнери можуть бути обмежені таким чином, щоб будь-які привілеї користувача root не обов'язково відповідали привілеям користувача root на хості.

- Стандартні базові образи ОС

Контейнерні рішення значно виграють від можливості створювати індивідуальні контейнери зі стандартних, широко використовуваних образів ОС. Ці стандартні образи отримують вигоду від постійного аудиту безпеки та виправлень, що робить їх надійним підґрунтям для створення індивідуальних контейнерів.

- Гнучкість і простота в налаштуванні

Контейнери легко налаштовуються і надають системним адміністраторам гнучкий контроль. Можна обмежити все, від привілеїв до runtime сервісів, що дозволяє обмежити доступ до ресурсів для будь-якого даного контейнера.

Однак, варто не забувати, що впровадження нових технологій в системи завжди буде пов'язане з додатковими ризиками для безпеки.

1.4 Тісно пов’язані проекти

Проекти, описані в цьому розділі, вивчаються з точки зору їх найцікавіших характеристик, а не досягнення відповідних цілей дослідження.

1.4.1 Адаптивні приманки

Цілі розгортання приманок можуть сильно відрізнятися, і більшість з них пропонують широкий вибір опцій і різних функцій для забезпечення гнучкості в цьому відношенні. Як вже обговорювалось раніше, розширення можливостей

взаємодії приманки призводить до отримання більш широкої та докладної інформації, а також до більшого потенційного збитку для системи. Різні компроміси в проектуванні приманок явно більш підходять для одних сценаріїв розгортання, ніж для інших.

На момент написання цієї роботи, на GitHub було публічно розміщено понад 1000 проектів приманок з відкритим вихідним кодом, які варіюються від виробничих приманок до великих спільних проектів і хобі-проектів. Дослідницькі приманки, які обговорювались в журналах і документах різних конференцій, ще збільшують це число, тому немає жодних сумнівів, що існує безліч запатентованих виробничих рішень [39].

Деякі найбільш значні проекти Honeypots, з якими було ознайомлено в ході цього дослідження, описані нижче. До них відносяться дослідницькі приманки, а також розробки від спільнот (community-based).

1) Kippo

Kippo – це приманка із середнім рівнем взаємодії та відкритим вихідним кодом, яка більше не знаходиться в активній розробці. Вона була розроблена спільнотою онлайн-розробників і призначена для захоплення всієї сеансової взаємодії зловмисника [47]. Kippo емулює Debian Linux і надає широкий спектр можливостей:

- Kippo надає готову файлову систему й оболонку для взаємодії. Кількість одночасних сесій є необмеженою. Також є можливість незалежно надавати підроблену файлову систему та оболонку кожному зловмиснику.
- Kippo реалізує фальсифіковану службу SSH, до якої зловмисники можуть підключатися.
- Kippo включає параметри конфігурації для дозволених комбінацій імен користувачів і паролів, які можуть контролюватися користувачем.
- Всі взаємодії з Kippo реєструються, включаючи спроби входу в систему методом «грубої сили», IP-адреси джерела, інформацію про протокол зловмисника, виконані команди та записи будь-яких спроб завантаження.

Kippo реалізований як додаток Python, тому зловмисники не взаємодіють безпосередньо з базовою ОС, що надає додатковий рівень захисту для хост-системи. Також він не потребує будь-якого зовнішнього програмного забезпечення для своїх основних операцій, що робить його менш вразливим для сторонніх ризиків.

Kippo не надає можливості запускати шкідливі програми, але може використовуватися разом з іншими рішеннями, такими як Cuckoo Sandbox – системою аналізу зловмисних програм з відкритим кодом, щоб безпечно виконувати та аналізувати шкідливі програми в контрольованому середовищі. [48]

На жаль, через те, що Kippo емулює реальну систему, він є прикладом приманки, чутливої до зняття відбитків системи: емулюючи сервіс OpenSSH можна ініціювати характерні відповіді, що ідентифікують пристрій як приманку Kippo, використовуючи ретельно створені повідомлення [37]. Цей недолік призвів до припинення активної розробки, що зробило Kippo значною мірою непридатним для розробки нових рішень на його основі.

2) Cowrie

Cowrie – це приманка із середнім рівнем взаємодії, яку було створено дослідником інформаційної безпеки Мішелем Остергофом на основі Kippo. Вона активно впроваджується і підтримується спеціалізованою спільнотою, яка забезпечує швидкий відгук на запити та виправлення помилок. Cowrie широко використовується в дослідницьких і виробничих середовищах [49];[50];[51].

На додаток до можливостей, що надаються приманкою Kippo, Cowrie володіє деякими додатковими функціями [52];[53]:

- Cowrie дозволяє зловмисникам отримати доступ за допомогою емульованих клієнтів telnet та SSH, підтримуючи два протоколи, які найбільш широко використовуються IoT ботнетами [20];[54].

- Cowrie записує всю інформацію про сеанс атаки в форматі JSON, який широко використовується і може оброблятися великою кількістю інструментів;
- Cowrie забезпечує інтеграцію з рядом корисних інструментів, включаючи базу даних шкідливих програм VirusTotal і приманку Mailoney SMTP;
- Cowrie також значно поліпшив свого попередника Kippo, вирішуючи більшість основних проблем зняття відбитків системи.

Цікавим є факт, що є у розробників Cowrie спостерігається деяка активність в сторону контейнеризації [55]. Це ще раз ілюструє зростаюче усвідомлення необхідності перенесення додатків безпеки в архітектури, керовані контейнером.

3) Honeyd

Honeyd є важливим попередником багатьох сьогоднішніх приманок і, ймовірно, є найвідомішою їх реалізацією. Він являє собою віртуалізовану приманку з відкритим вихідним кодом, розроблену Ніельсом Провосом у 2002 році, яка використовується по сьогоднішній день.

Honeyd є гнучкою та адаптивною в тому сенсі, що вона може бути налаштована як для запуску довільних сервісів, так і для роботи під управлінням різних операційних систем, вивчаючи «особистість» сервісу та читаючи файли Nmap, щоб імітувати їх [56]. Однак вона не може емулювати системні компоненти, такі як файлові системи або виконання команд. Як пояснено в статті розробника, Honeyd «слухає мережеві запити, призначені для її сконфігурованих віртуальних приманок» і «відповідає відповідно до служб, які працюють на віртуальному Honeypots. Перед відправленням відповідного пакета в мережу цей пакет модифікується механізмом ідентифікації Honeyd, щоб відповідати поведінці мережі налаштованої операційної системи» [57].

Додаткові функції Honeyd охоплюють можливість створення топології віртуальної маршрутизації та можливість поліпшення налаштованих маршрутів з реалістичними характеристиками затримки й втрати пакетів, щоб здаватися більш переконливими. Хоч ця приманка послугувала джерелом натхнення для

пошуку адаптивних рішень, той факт, що вона більше не розробляється, робить її значною мірою непридатною для використання в нових розгортаннях.

4) IoTPot

IoTPot – це дослідницька приманка, розроблена Інь Мінн Па Па та ін., яку вони називають «новітньою приманкою, яка емулює взаємодію протоколу telnet і різних пристроїв IoT» [58]. Це реагуюча IoT-приманка, призначена для вивчення атак на основі Telnet. Особливий інтерес полягає в її здатності реагувати на запити, надіслані зловмисниками, надаючи їм динамічно згенеровану відповідь, яка, як вважається, найбільш ймовірно відповідає архітектурі ЦП системи, на яку націлена атака. Схема архітектури приманки, яку було взято зі статті розробників IoTPot, зображена на рисунку 1.7 [58]:

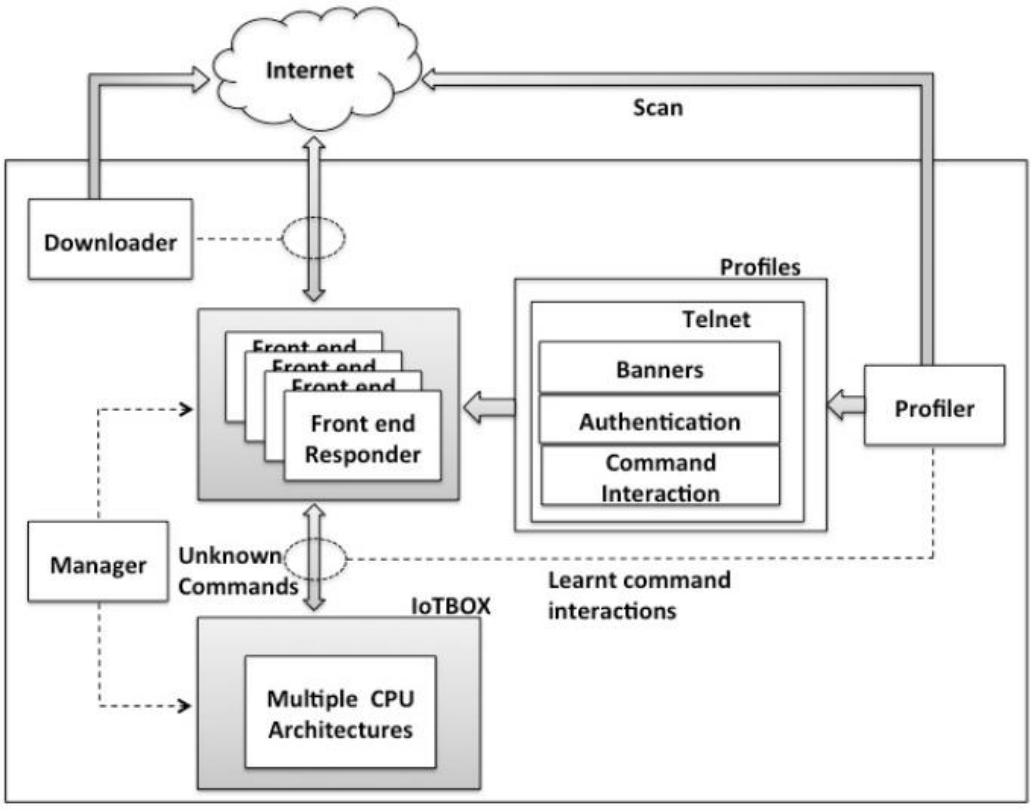

Рисунок 1.8 – Архітектура приманки IoTPot. Оригінальне джерело: The Cowrie Project, GitHub [55].

IoTPot була розроблена з урахуванням того факту, що існує надто багато різних типів пристроїв IoT, щоб була можливість розробити приманку для них всіх. Принципи її роботи значною мірою засновані на мережі приманок SGNET,

в якій датчики з низьким рівнем взаємодії направляють атаки, які вони отримують, на приманку з високим рівнем взаємодії для обробки [59]. Це схоже на функцію зовнішнього відповідача в IoTPot, який є компонентом системи безпосередньої взаємодії зі зловмисниками.

IoTPot, як адаптивна telnet приманка**,** здається дуже ефективною. Проте, вона є відносно обмеженою, оскільки може мати справу тільки з атаками на основі telnet, обмежуючи типи атак, які може захоплювати.

5) IoTCandyJar

IoTCandyJar – це недавно створена IoT приманка, яка класифікується розробником як приманка інтелектуальної взаємодії, а не рівня (низького, середнього чи високого) взаємодії [60]. Дослідники пояснюють, що: «мета інтелектуальної взаємодії полягає в тому, щоб навчитися «правильній» поведінці для взаємодії з клієнтами з нульовими знаннями про пристрої IoT». Підхід розробника полягає у використанні методів машинного навчання для визначення найбільш відповідної тактики для продовження атак.

IoTCandyJar складається з декількох основних модулів, які показані на рисунку 1.9:

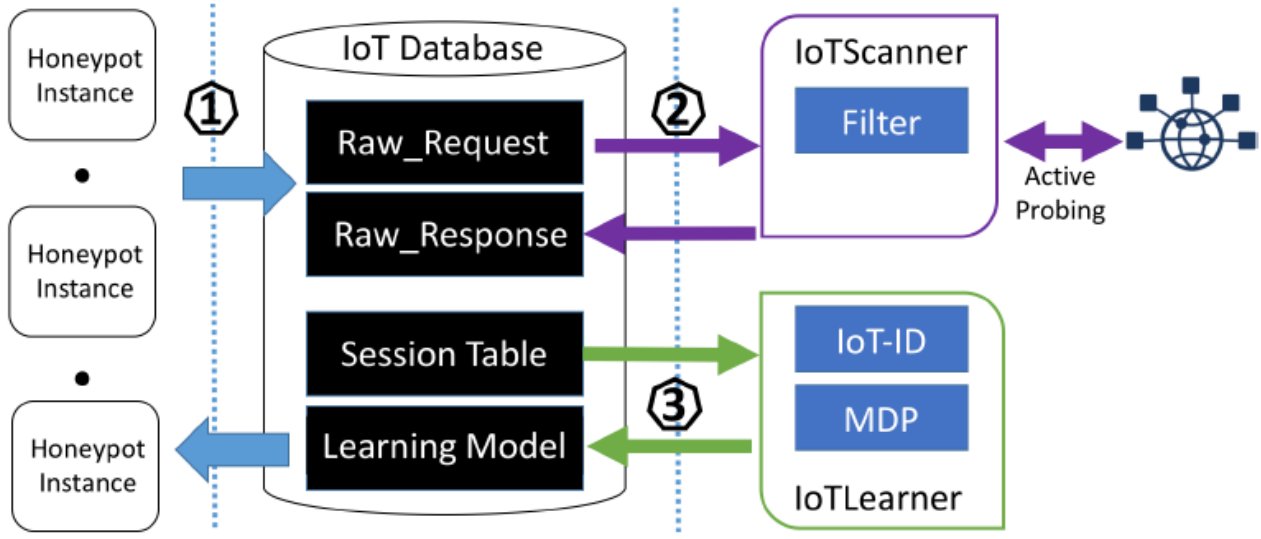

Рисунок 1.9 – Архітектура приманки IoTCandyJar. Оригінальне джерело: IoTCandyJar: Towards an IntelligentInteraction Honeypot for IoT Devices [60].

- Модуль *IoTScanner* використовує той факт, що для ідентифікації пристроїв IoT в Інтернеті вільно доступні потужні інструменти сканування (Shodan, Censys, Masscan і Zoomeye). За допомогою цих інструментів він активно досліджує ці пристрої, збираючи їх відповіді на різні запити, які раніше були перехоплені IoTCandyJar.
- Модуль *IoTLearner* вбудовує в приманку інтелект, використовуючи відповіді, отримані модулем *IoTScanner*, для навчання моделі з використанням алгоритмів машинного навчання. Точність відповідей на запити підвищується з ростом числа взаємодій зі зловмисниками. Якщо зловмисник відправляє другий запит після отримання відповіді від IoTCandyJar, відповідь вважається правильною.
- *IoTDatabase* фіксує всі захоплені запити/відповіді та зберігає будь-яку інформацію, що стосується моделі машинного навчання.

Основна особливість, на яку наголошують розробники IoTCandyJar – це динамічний підхід до реагування на зондування ботнетами IoT. Подібно IoTPot, IoTCandyJar прагне забезпечити найкращу відповідь зловмисникові на основі того, що він має намір шукати. Основна відмінність між цими двома приманками полягає в підході, який вони використовують для досягнення цієї мети: IoTPot визначає найбільш прийнятну відповідь генеруючи правильну відповідь за допомогою виконання запиту, IoTCandyJar дізнається відповідь на запит, заснований на взаємодії зі зловмисниками. Вона генерується виключно на основі того, що система дізналася з попередніх взаємодій, а не шляхом будь-якого виконання запитів.

6) Zigbee Honeypot

Zigbee це дослідницька приманка, розроблена Даулінгом та ін., і призначена для Zigbee пристроїв, які зазвичай використовуються в WSN [14]. Розробка цієї приманки була мотивована тим фактом, що, оскільки ці пристрої IoT використовуються більш широко, то і їх уразливості також стають зрозумілішими, а, отже, оцінка загрози для цих пристроїв дуже важлива.

У своїй реалізації приманки, Даулінг та ін. включили ряд цікавих заходів для ініціювання інтересу зловмисників до приманки:

- Ім'я хоста для honeypot було налаштоване як zigbee-gateway як засіб оцінки того, чи існує яка-небудь обізнаність про пристрої Zigbee серед зловмисників;
- Дослідники впровадили засоби передачі фіктивного медичного трафіку, який передається в не зашифрованому вигляді по не фільтрованій мережі.

Генерування потенційно цікавого мережевого трафіку для того, щоб спонукати зловмисників атакувати приманку, змушує задуматися про це рішення. Хоча проведене дослідження в основному сфокусоване на виявленні будь-якої обізнаності про пристрої Zigbee, ідея адаптації приманки до атакуючих, яких вона хоче залучити, є корисною.

1.4.2 Контейнерні приманки

Число пов'язаних проектів, визначених в області контейнеризації приманок, обмежене за кількістю головним чином тому, що домен безпеки все ще наздоганяє міграцію інфраструктур на архітектури на основі контейнерів. Дослідження, проведені в цих суміжних проектах, фокусуються як на переваги використання контейнерів для розміщення приманок, так і на проблеми, з якими вони при цьому зіштовхуються.

У 2014 році П. Пісарчік та ін. розробили розподілену мережу приманок (Honeynet) з високим рівнем взаємодії з використанням контейнерів для віртуалізації на рівні операційної системи, підходу, який в той час був відносно не досліджений [61]. Мотивація для розробки такого рішення була наступною:

- Той факт, що адміністрування систем, керованих приманкою, потребує багато часу, мотивуючи використання контейнерів для спрощення їх управління;

- Можливість приманок зіставляти події атаки, щоб визначити, чи є вони локалізованими або розподіленими, швидко дозволяючи ідентифікувати їх тенденції.

Хоча підхід до розробки мережі Honeynet заслуговує на увагу, більш значний вклад, внесений цим дослідженням, пов'язаний з контейнеризацією приманок. Він підкреслює, що в порівнянні з віртуальними машинами або bare-metal системами, віртуалізація на рівні операційної системи вимагає дуже мало продуктивності та витрат на обслуговування [61]. Крім того, дослідники відзначають, що використання контейнерів надає додатковий рівень оманливості: оскільки приманки є ізольованими середовищами, які спільно використовують ядро реальної ОС, вони з більшою ймовірністю будуть виглядати як реальна система при знятті відбитків системи.

Всі ці функції створюють враження, що контейнери являють собою ідеальне рішення для розгортання приманок. Крім цього, вони також усувають потенційні помилки безпеки при використанні контейнерів для розміщення, особливо ризики для хост-системи, пов’язані з поділом ядра ОС з контейнерами.

В роботі, опублікованій у 2017 році, Кедрович та ін. досліджують використання контейнерів на основі Linux (LXC) в розгортанні приманок [62]. Однак, використання LXC тут розглядається з точки зору їх здатності ухилятися від виявлення змін своєї поведінки шкідливим ПЗ після викриття середовища на основі віртуальних машин [63]. Мотивація полягала в тому, щоб визначити, чи будуть контейнерні середовища здійсненними в довгостроковій перспективі як підхід до розміщення приманок без їх виявлення шкідливим ПЗ.

Цими ж дослідниками було проведено ряд експериментів, в яких аналізувались слабкі сторони LXC, пов'язані з цифровими відбитками стосовно слабких систем і віртуальних машин. Вони сфокусувалися на типових методах, які шкідливі програми використовують в віртуальних машинах: порівняння відгуку, що генерується для конкретного введення, з відгуком, який можна

очікувати від системи з «голим залізом». Зокрема, було вирішено протестувати та порівняти такі властивості віртуальних машин і контейнерів:

- Мінливість і час виконання в дискретизації тактової частоти процесора;
- Інформація про процесор, яка повідомляється;
- Час виконання інструкції.

При оцінці їх експериментів було виявлено, що контейнери були набагато менш сприйнятливі до зняття цифрових відбитків, ніж віртуальні машини, що й очікувалося через той факт, що LXC виконуються безпосередньо на апаратних засобах з «голим залізом» як ізольовані процеси, тоді як віртуальні машини – ні. Однак було рішуче зроблено висновок про те, що, якщо розробник шкідливого програмного забезпечення захоче визначити контейнерне середовище за допомогою засобу зняття відбитків системи, то у нього не виникне ніяких складнощів.

1.4.3. Монітори кібер-інцидентів, керовані приманками

Проекти, які розглядаються далі, є актуальними, оскільки їх використання робить приманки більш доступним як для окремих осіб, так і організацій в цілому. Вони визнають важливість агрегування і візуалізації даних з декількох приманок, щоб надавати важливі дані ключовим особам, які приймають рішення, заохочуючи використання активного захисту мережі та спрощуючи управління.

1) TraCINg

TraCINg – це монітор кіберінцидентів, який керується приманкою і розроблений дослідником Е. Василоманолакіс та ін. у 2015 році [43]. Його розробка була мотивована спостереженням того, що консолідація даних, зібраних з приманок в різних контекстах розгортання, може дозволити корелювати дані атак, дозволяючи ідентифікувати спалахи пов'язаних атак, які виникають.

Система TraCINg збирає дані з великого числа географічно розподілених приманок з метою кореляції подій атаки. Багато з цих приманок розгорнуті на хмарних платформах, яке було обране дослідниками для забезпечення найбільшої розмаїтості місць розгортання і максимального часу безвідмовної роботи системи для безперервного моніторингу. Довільні приманки з відкритим вихідним кодом можуть використовуватися з системою TraCINg за умови, що їх дані записані в форматі JSON.

Це рішення для моніторингу інцидентів не розглядається в виробничих системах, оскільки воно не призначене для високого ступеня розгортання, але в ньому розглядається питання про агрегацію і підсумовування складних даних осмисленим чином за допомогою візуалізації, що дозволяє робити короткий опис важливих даних для ключових осіб, що приймають рішення. Пропозиція розробників щодо кореляції даних з великих розгортань розподілених приманок дуже цікава, однак, в рішенні використовуються дві приманки з низьким рівнем взаємодії, що обмежує рівень деталізації захоплених даних атаки. Удосконалена система зможе збирати більше даних про події атаки, щоб краще зрозуміти мотиви і методи атакуючих.

2) TPot

Проект TPot – це монітор інцидентів на основі приманок, який розроблено і використовується Deutsche Telekom. Його створення пояснюється тим, що фахівці компанії вирішили використовувати власні розробки з відкритим вихідним кодом, щоб зробити розгортання приманок у виробничих мережах більш доступним [50].

TPot підтримує кілька приманок з відкритим вихідним кодом, розміщених в контейнерах. Система складається з ряду компонентів, найбільш значущі з яких перераховані нижче:

- Контейнерні приманки

Система TPot використовує переваги контейнерів, перераховані в розділі 1.3.2, для забезпечення високого ступеня розгортання системи. Контейнерні

приманки, які підтримуються платформою TPot, активно допрацьовуються розробниками й призначені для різних рівнів інтерактивності та атакуючих.

- The ELK Stack

Стек Elasticsearch-Logstash-Kibana (ELK) являє собою набір інструментів управління журналами, які виконують функції пошуку, ведення записів та візуалізації.

Інфографіку, надану розробниками TPot, можна побачити на рисунку 1.10. Кожна приманка визначається Dockerfile, який вбудований в образ. Потім цей образ використовується для запуску приманки, і дані, що нею генеруються, зберігаються в томах, які спільно використовуються з хост-системою. Згодом дані можуть бути передані в інші системи, такі як Deutsche Telekom EWSPoster, як показано на рисунку.

Всі образи контейнерів для приманок побудовані на основі образу контейнера Alpine Linux, який має особливо маленький розмір, що означає, що його обсяг зберігання є мінімальним. Перевага використання приманок всередині контейнерів, яку розробники виділяють на своїй веб-сторінці, полягає в тому, що вони можуть «запускати кілька приманок на одному і тому ж мережевому інтерфейсі, зберігаючи при цьому невелику площу й обмежуючи кожну приманку у своєму власному середовищі» [64].

Пропозиція щодо використання переваг контейнерів для розміщення приманок є інтригуючою, особливо з урахуванням того, що труднощі розгортання та обслуговування приманок на сьогодні є величезною перешкодою для їх широкого поширення. Робота Deutsche Telekom над цим проектом підсилює аргумент на користь об'єднання контейнерних розгортань приманок і інструментів візуалізації.

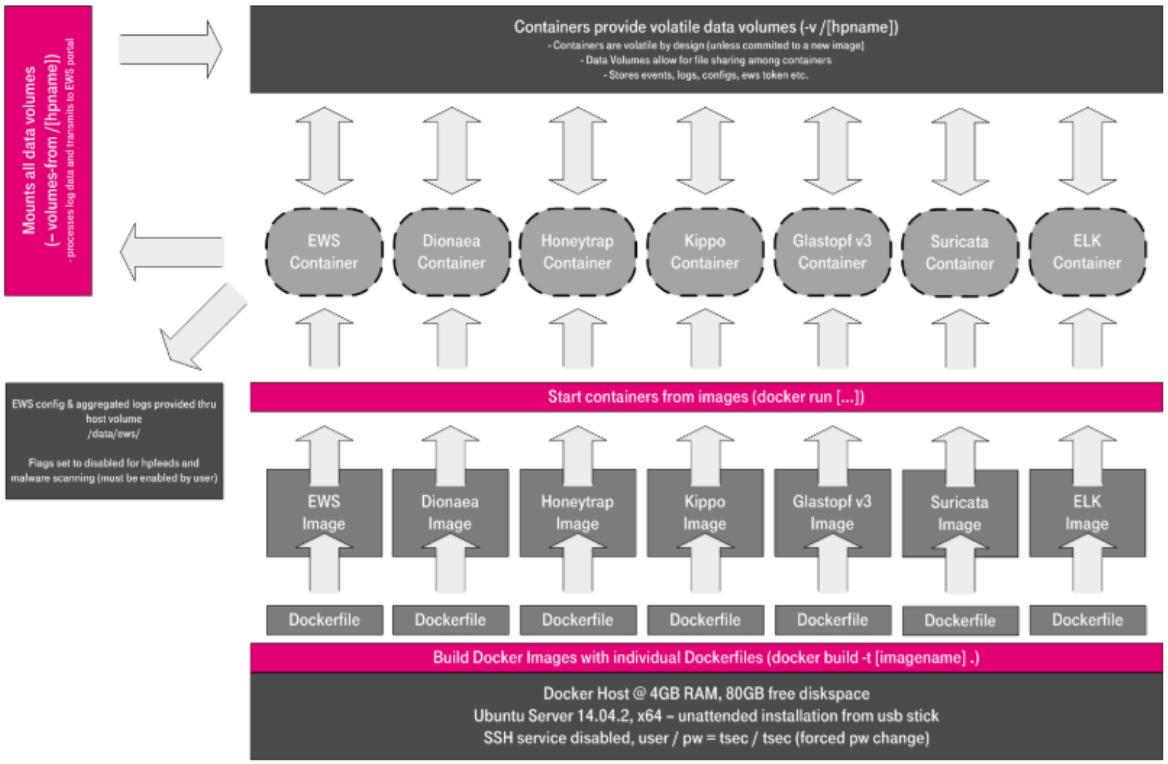

Рисунок 1.10 – Діаграма архітектури TPot. Оригінальне джерело: DTAG Community Honeypot Project [64].

3) Modern Honey Network

Modern Honey Network (MHN) – це виробнича система, розроблена компанією Anomali Inc. для управління розгортанням корпоративних приманок. Їх рішення відкрити систему з відкритим вихідним кодом було мотивовано тими ж причинами, що і розробників TPot, які полягають в тому, що розгортання та обслуговування розгортання приманок завжди було «складним процесом, призначеним для компаній по забезпеченню безпеки та дослідників в галузі безпеки» [51].

MHN – це централізована серверна система, яка займається управлінням розгортання приманок, а також агрегацією їх даних. Вона підтримує велику кількість приманок з відкритим вихідним кодом, включаючи Cowrie. Всі ці приманки – це приманки з відкритим вихідним кодом з низьким або середнім рівнем взаємодії, які, за словами команди MHN, були обрані, щоб мінімізувати ризик, пов'язаний з розгортанням у виробничому середовищі. Дані приманок, які збираються системою MHN, можна візуалізувати за допомогою Splunk –

інструменту аналізу та візуалізації даних. Однак, на відміну від підтримки TPot для стека ELK, Splunk не поставляється як частина системи MHN.

1.5 Формулювання проблеми

Як детально обговорювалося в попередніх розділах, швидке просування загроз безпеки у відповідь на технологічні розробки та інновації робить надзвичайно важливим, щоб методи та інструменти пом'якшення не відставали. Цілком очевидно одне: існує нагальна потреба в переоцінці поточної передової практики в області кібербезпеки. Особливо це стосується об’єктів критичної інфраструктури, в яких відсутність реалізованих механізмів безпеки представляє величезну небезпеку. Саме це міркування лежить в основі формулювання цілей даного дослідження.

1.5.1 Виявлені проблеми

У попередніх розділах було розглянуто ряд проблем, що стоять в даний час перед безпекою сучасних інфраструктур і систем. Завдання, які, як виявилося, становлять найбільший інтерес для цього дослідження, викладено далі.

Як пояснюється в контексті атак вимагачів WannaCry на NHS в розділі 1.1.3, IT системи у багатьох організаціях можна порівняти з системою «Black-box», ілюстрацію якого можна побачити на рисунку 1.11. В систему «чорного ящика» передається деяка відома вхідна функція f(x). Далі над нею виконуються невідомі операції, і в результаті генерується нова вихідна функція g(x).

Така система не дає ніяких вказівок на те, що відбувається всередині, поки вона не починає поводитись несподівано, а це означає, що при виникненні проблем їх важко діагностувати й майже неможливо вирішити. Проте, неефективні механізми зворотного зв'язку часто настільки погані, що не мають

ніякого чинного механізму: пасивні механізми, такі як брандмауери, неточні у своїй підозрі на зловмисну діяльність, а це означає, що значна подія може легко залишитися непоміченою [31]. Надзвичайно важливо використовувати ефективні механізми зворотного зв'язку, які будуть простими у використанні та здатними передавати ключову інформацію тим, хто діятиме відповідно до неї.

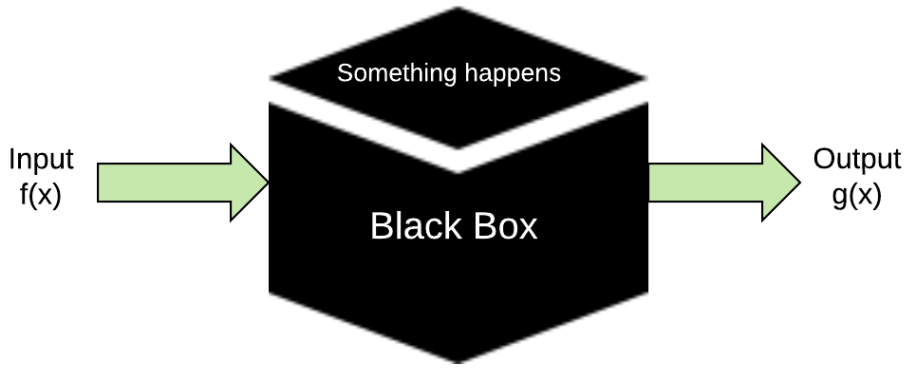

Рисунок 1.11 – Ілюстрація системи Black-Box

Очевидно, що для реалізації ефективних механізмів протидії загрозам спочатку необхідно зрозуміти найбільш важливі ризики для системи, дещо, що було підкреслено в інтерв'ю зі Стюартом Рідом, старшим директором з безпеки NTT, який стверджував, що «найкраща відправна точка це зрозуміти, звідки... ризики приходять» [65]. Це мотивує і наголошує на необхідності в технології, яка здатна надавати інформацію про ці активні, динамічні загрози.

Як обговорювалося в розділі 1.2.2, хоча використання пасивних технологій виявлення вторгнень, таких як брандмауери, займає важливе місце в захисті IT систем, вони мають обмежені діагностичні можливості, оскільки здатні протистояти тільки відомим загрозам. В аналогії, яка порівнює кібербезпеку зі здоров'ям, Фред Шнайдер пояснює, що «як і в випадку з медичними проблемами, деякі атаки найкраще вирішувати відразу ж» [34].

Рішення на основі приманок є перспективним в тому сенсі, що саме вони можуть запропонувати відносно активного захисту мережі та швидкого реагування на інциденти. Сучасні IT інфраструктури гостро потребують таких

адаптивних підходів до захисту: захисних механізмах, які, попри невизначеність щодо характеру атак, можуть продовжувати забезпечувати зниження загрози. З обговорення, представленого в розділі 1.2.3, зрозуміло, що рішення Honeypots здатні задовольнити цю вимогу.

- Інформація, яку можуть захопити honeypots, дозволяє постійно покращувати структуру і захист інфраструктури;
- Низька кількість помилкових спрацьовувань, робить виявлення загроз простим: всі зареєстровані події, як правило, можна вважати погрозами;
- Honeypots здатні оповіщати про загрози**,** як тільки вони виникають, що дозволяє реагувати до того, як атака зможе поширитися.

Проте, як було визначено в розділі 1.4.3, ці переваги виявляються в знак визнання того факту, що широке розгортання систем на основі honeypots до теперішнього часу було ускладнене через накладні витрат на обслуговування, які вони несуть системним адміністраторам. Щоб така система була практичною і здійсненною для розгортання у виробничому середовищі, вона повинна бути:

- Недорогою в розгортанні, експлуатації і обслуговуванні;
- Ефективною та надійною у захисті мережі;
- Простою у використанні

Щоб бути придатним для широкого поширення в якості служби безпеки, рішення для розгортання на основі honeypots має відповідати всім цим вимогам.

Як видно з тісно пов’язаних проектів, розглянутих в розділі 1.4, в проектуванні приманок досягнутий значний прогрес. Однак, до сих пір існує мало переконливих досліджень, які стосуються поліпшення проектування приманки для того, щоб викликати інтерес зловмисників.

Майже у всіх розглянутих дослідженнях основна увага приділялася заохоченню зловмисників до тривалої взаємодії з приманками, а не залученню до цієї взаємодії в першу чергу [53];[58];[60];[14]**.** С. Даулінг та ін. досліджували деякі цікаві заходи щоб спонукати зловмисників, які намагаються

скомпрометувати пристрої Zigbee, але вплив цих заходів на число скоєних атак не представляв інтересу для кінцевого дослідження [14].

Ефективність технологій на основі honeypots в захисті систем значною мірою залежить від їх здатності відводити атаки від цінних систем в середовищі, яке вони захищають. Розуміння того, як honeypots можна спроектувати так, щоб вони були настільки привабливими, наскільки це можливо для зловмисників, підвищило б їх адаптивність у швидкому оповіщенні перших респондентів про той факт, що вони були атаковані, що дозволяє вжити заходів до того, як наступна атака буде націлена на цінні системи.

1.5.2 Пропоноване рішення

Зрозуміло, що активний підхід до зниження ризиків для об’єктів критичних інфраструктур – це крок вперед, і що можливості honeypots можуть істотно допомогти в досягненні цього. Хоча існує ряд підходів як в області досліджень, так і у виробництві, які описані в розділі 1.4, до сих пір немає єдиного рішення, яке могло б забезпечити практичні та здійсненні засоби використання активного захисту мережі як єдиного розгорнутого пристрою.

З огляду на проблеми, визначені в цьому розділі**,** щодо їх вирішення було сформульовані два завдання дослідження.

Завдання 1 – моніторинг кіберінцидентів для об’єктів критичної інфраструктури. Воно спрямоване на те, щоб задовольнити потребу в можливому активному вирішенні для виявлення вторгнень, що забезпечує швидке реагування, дозволяючи вживати заходів щодо загроз з мінімальними накладними витратами.

Вимоги такого рішення охоплюють:

1) Економічну ефективність, швидкість в розгортанні та масштабуванні, сумісність з існуючою IT інфраструктурою.

2) Наявність ряд дуже привабливих приманок, які можуть ефективно відвернути зловмисників від цінних машин в мережі, в якій вони розгорнуті.

3) Точка централізації для даних, зібраних з будь-яких розгорнутих приманок, щоб адміністратори мали можливість отримати цілісне уявлення про свою систему.

4) Ефективна і надійна система оповіщень, щоб ключові сторони могли бути попереджені якомога швидше після того, як система виявить подію загрози.

Складові компоненти такої системи розглядаються в наступних розділах.

Як докладно описано в розділі 1.2.3, приманки дозволяють ідентифікувати вразливості та недоліки в системі, в якій вони розгорнуті, і, на відміну від механізмів пасивного захисту, здатні пом'якшувати ризики, які раніше були невідомі. Це робить їх очевидним вибором для реалізації активного захисту мережі в об’єктах критичної інфраструктури.

У запропонованій системі моніторингу інцидентів використання ряду приманок в конфігурації з мережею було визначено як ідеальний підхід до забезпечення ефективного моніторингу. У підходу на основі honeynet є кілька переваг: можливість вивчення поширення атаки через мережу, а також збільшена поверхня атаки для відволікання зловмисника від цінних систем в тому ж середовищі. Приманки в запропонованій системі діятимуть як датчики, забезпечуючи механізм зворотного зв'язку щодо подій загроз в мережі, та реєструючи дії атак для аналізу.

Враховуючи властивості контейнерів, які були описані в розділі 1.3.2, їх було визначено як ефективний варіант розгортання для пропонованої системи. Розгортаючи приманки в міжмережевих контейнерах, стає можливим забезпечити гнучке і недороге рішення для розгортання, яке не потребує обслуговування. Один контейнер буде використовувати відносно невелику частку потужності процесора і пам'яті комп'ютера, а кілька інших на одному хості зможуть використовувати один і той же фізичний мережевий інтерфейс, водночас реалізуючи свій власний незалежний мережевий стек [62]. Важливо

відзначити, що будь-який контейнер може бути розгорнутий і видалений з мінімальними експлуатаційними витратами через його переносні конфігурації. У випадку об’єктів критичних інфраструктур, використання такої архітектури дозволило б відновити будь-який контейнерний компонент системи після компрометації з мінімальним часом простою.

З точки зору накладних витрат і обслуговування, контейнери є відмінним вибором як платформи для розміщення додатків в сучасній IT інфраструктурі. Міграція додатків безпеки на контейнерну архітектуру вигідна і важлива для організацій, що розміщують свою інфраструктуру в контейнерах: успіх контейнерних приманок на платформі TPot ілюструє це [50]. Крім того, контейнери забезпечують ступінь відтворюваності, який зазвичай є недосяжним у дослідженнях в області комп'ютерних наук, як зазначає Ю. Сіто та ін. [66]. Всі ці переваги дають міцну основу для використання контейнерів при розміщенні рішення на основі приманок.

Аналіз даних приманки значно спрощується завдяки використанню інструментів візуалізації, які дозволяють класифікувати та візуалізувати дані складних атак, щоб зробити значущі висновки.

Як частина рішення з моніторингу інцидентів на основі приманок, зрозуміло, що отримана з них інформація виграє від агрегування, аналізу і візуалізації в центральній системі управління, яка консолідує дані та виконує операції з ними, перш ніж зробити їх доступними для користувача. Проекти MHN і TPot є прикладом такого підходу [50];[51].

Така система управління є найважливішим компонентом моніторингу кіберінцидентів, в якому дані про атаки збираються централізовано. Окрім цього, вона повинна бути безпосередньо доступна для приманок, але прихована від зловмисників.

Проблеми, які виникли після інфікування систем NHS програмами-вимагачами WannaCry, продемонстрували важливість наявності механізму зворотного зв'язку в разі виявлення небажаної поведінки. Ефективний монітор

кіберінцидентів повинен забезпечувати цей механізм, який дозволить представляти дані про загрозу тим, хто зможе на них впливати.

Завдання 2 – пропозиція щодо адаптивних приманок. Зрозуміло, що ефективність приманок при виявленні вторгнень значною мірою залежить від їх конструкції, яка, у свою чергу, залежить від контексту їх розгортання і цінності, яку вони повинні надавати. Єдина точка дотику між усіма розгортаннями приманок полягає в тому, що вони повинні бути адаптивними, щоб ефективно обманювати зловмисників, допомагаючи їм в тому, що, на їхню думку, є успішним розвитком їх атаки, і водночас спокійно стежити за їх діями.

Ця потреба в адаптивності починає все частіше розглядатися в сучасних дослідженнях, але до сих пір є прогалини, які дають можливість для поліпшення. Зокрема, всі тісно пов'язані між собою проекти приманок, визначені в розділі 1.4, зосереджені на підходах, спрямованих на підтримку інтересу зловмисників, а не на ініціювання інтересу в першу чергу [53];[58];[60]. Розробка адаптивних приманок таким чином, щоб вони рекламували найбільш популярні характеристики систем-жертв, являє собою великий інтерес.

Проведення ретельної оцінки ефективності окремих характеристик приманки для залучення інтересу зловмисників є цікавим, але складним завданням. Ефективність приманки складно виміряти кількісно, але зрозуміло, що ефективна приманка забезпечить ефективне виявлення вторгнень і, в залежності від контексту її розгортання, збере відповідні дані атак з певним рівнем деталізації.

Ефективність приманки потенційно може бути виміряна за допомогою таких властивостей, як кількість залучених атак, кількість виявлених справжніх спрацьовувань, здатність визначати мотивацію зловмисника на основі отриманих даних і т.п. [37]. Вірогідний вимір цих властивостей значною мірою залежить від підтримки узгодженого дослідницького середовища, в якому можна точно оцінити відносну ефективність.

Honeynet - це ефективний спосіб точного і одночасного вимірювання привабливості певних характеристик приманок в експериментах, оскільки в одному і тому ж середовищі можна випробувати кілька характеристик протягом одного і того ж періоду часу. Використовуючи конфігурацію honeynet в дослідницькому середовищі, можна збирати дані з декількох приманок при однакових умовах середовища, і вимірювати відносну продуктивність на основі конфігурацій, які дещо відрізняються.

Із самого початку дослідження однією з найбільш цікавих і активних категорій ботнетів була мережа ботнетів IoT, революційне і недавнє явище, що викликає серйозну стурбованість. Пропозиція підходу до розробки адаптивних приманок з акцентом на те, щоб зробити їх максимально привабливими для бот-мереж IoT, сприяло б підвищенню рівня обізнаності про цих зловмисників, завдяки розумінню їх бажаних вразливостей і цільових систем. Конфігураційні та інформаційні можливості приманок роблять їх привабливим варіантом в цьому відношенні. Тому орієнтація на цих зловмисників при розробці та впровадженні пропонованої системи стане цінним додатковим внеском в область інформаційної безпеки.

1.6 Висновок до розділу

Виходячи з теоретичного підґрунтя, представленого в цьому розділі, очевидно, що область кібербезпеки динамічна і сповнена проблем. Як влучно підмітив Фред Шнайдер: «Як і хороше здоров'я, кібербезпека ніколи не буде вирішуваною проблемою» [55].

Багато з цих проблем є результатом грубої недбалості з боку виробників і розробників систем, очевидні недогляди і помилки внаслідок недостатньої зручності використання. Інші ж є результатом нерозуміння наслідків для конфіденційності та безпеки, пов’язаних з використанням небезпечних технологій. Все це представляє можливість для поліпшення сучасних підходів до впровадження технологій і пов'язаних систем.

Робота, виконана дослідниками та розробниками, які стоять за тісно пов'язаними проектами, описаними в розділі 1.4, вказує на існуючу обізнаність та інтерес щодо розв'язання проблем, з якими зіштовхуються в області безпеки сучасних систем. Подібний інтерес і обізнаність мотивують рішення поліпшити поточний стан справ в цій галузі.

2 ПОЧАТКОВІ ПРОЕКТНІ РІШЕННЯ З РОЗРОБКИ АДАПТИВНИХ КОНТЕЙНЕРНИХ ПРИМАНОК З АКТИВНИМ МОНІТОРИНГОМ

Для досягнення цілей дослідження, які викладено в попередньому розділі**,** при розробці пропонованої системи повинні бути враховані вимоги, визначені в розділі 1.5. Виходячи з цього, був розроблений початковий проект системи.

Існує безліч факторів, які необхідно враховувати при проектуванні дослідницького середовища, особливо коли некоректна структура може мати далекосяжні наслідки для безпеки інших людей, які в першу чергу не знають про проект. Для розробки рішень, які були б безпечними і ефективними, кожен аспект розробки вимагає уважного і ретельного планування.

2.1 Функціональні міркування

У цьому розділі описуються багато міркувань щодо функціонального проектування і рішень, прийнятих щодо реалізації дослідницького середовища. До них відносяться початкові рішення, що стосуються платформ розгортання середовища, конфігурації honeynet і проектування системи моніторингу.

2.1.1 Платформа для розгортання середовища

Вибір інфраструктури хостингу і платформ для пропонованої системи є важливим фактором при проектуванні, який справляв би значний вплив на успішну розробку монітора кіберінцидентів, а також на пропозицію по розробці адаптивних приманок в цілому.

Переваги послуг аутсорсингового хостингу сторонньої хмарної платформи докладно обговорювалися в розділі 1.3.1. Як пояснили Е. Васіломанолакіс та ін., «Хмарні сервіси забезпечують відмовостійкість і

надійність безвідмовної роботи..., що забезпечує безперервний моніторинг» [43].

Amazon Web Services (AWS) є дочірньою компанією Amazon Inc., яка позиціонує себе як «безпечну платформу хмарних сервісів, що пропонує обчислювальну потужність, сховище баз даних, доставлення контенту та інші функції, що допомагають компаніям масштабуватись і рости» [67].

Сервіс AWS Elastic Compute Cloud (EC2) є дуже гнучким засобом швидкого налаштування об'єктів сервера на вимогу. Попри те, що були розглянуті інші хостингові рішення, в тому числі інфраструктури OpenNebula від DigitalOcean, гнучкість варіантів, пропонованих AWS, а також той факт, що він є лідером в області надання хмарних послуг, привели до того, що в кінцевому підсумку він став платформою хостингу для цього дослідження.

Дійшовши висновку, що контейнери є найкращою платформою для розгортання приманок в запропонованій системі, було вирішено, що для реалізації дослідницького середовища буде використовуватися технологія контейнерів Docker. Docker є галузевим стандартом для контейнерних рішень і являє собою технологію з відкритим вихідним кодом та глобальною спільнотою підтримки.

Використання контейнерів Docker, зокрема для реалізації дослідницького середовища, має декілька переваг, зокрема:

- Кожен контейнер має незалежний мережевий стек, що спрощує реалізацію мережі honeynet в порівнянні з віртуальними мережами.
- Незалежність і гнучкість ізольованого контейнерного зберігання.
- Репозиторії Docker, використання яких означає, що строго перевірений образ базової ОС може бути використаний як основа для контейнерів приманок, а потім доповнений налаштованими параметрами конфігурації.
- Наявність досліджень, що стосуються безпеки технології Docker [68].
- Наявність вичерпної документації для екосистеми Docker, а також активна спільнота підтримки.

Всі ці фактори дозволять прийняти впевнене рішення щодо використання Docker в запропонованій системі.

Як для хостів віртуальних машин EC2, так і для образів контейнерів було обрано операційні системи на основі Linux, і цьому є ряд причин:

1) Linux вважається найбільш поширеною операційною системою для серверів і вбудованих пристроїв. Таким чином, націлювання рішення на розгортання в інфраструктурі на основі Linux, швидше за все, буде реальним сценарієм розгортання для такої системи.

2) Linux підтримується практично всіма інструментами з відкритим вихідним кодом і галузевими інструментами, які дозволять використовувати практично будь-яку технологію при реалізації проекту.

3) Велика частка кібератак припадає саме на Linux-системи.

У зв'язку з рівнем активності бот-мереж IoT було вирішено, що найбільш прийнятним вибором для цього проекту є приманка IoT. Приманка Cowrie, яка детально обговорювалась в розділі 1.4.1, була визначена як ідеальний варіант для націлювання ботнетів IoT в пропонованому розгортанні, особливо із урахуванням того, що вона є приманкою SSH/telnet, тобто протоколів, які, як було встановлено, найбільш активно використовуються в останніх IoT ботнетах [20];[54].

Той факт, що Cowrie є приманкою середнього рівня взаємодії, означає, що ризики компрометації реальної системи знижуються, а також забезпечується відносно високий рівень інтерактивності з зловмисником. Як проект з відкритим вихідним кодом, Cowrie також легко налаштовується, пропонуючи як високий ступінь конфігурованості, так і можливість розширення вихідного коду. Це є важливим фактором для успішного проведення експериментів з метою пропозиції ефективних підходів до проектування IoT-приманок.

Важливо відзначити, що Cowrie забезпечує високий рівень інтерактивності зі зловмисниками, одночасно знижуючи ризики компрометації

реальної системи, наприклад не даючи можливість виконувати код, замість цього зберігаючи контрольну суму SHA при будь-яких спробах завантаження. Окрім цього, Cowrie активно підтримується, і деякі початкові бачення контейнерного рішення для приманки вже існують.

Очевидно, що важливо враховувати, наскільки обмежені можливості контейнерів приманок для захоплення атак, а також для забезпечення безпеки систем. Якщо зловмисникові вдасться обійти додаток, він повинен бути обмежений в діях, які може виконувати при безпосередній взаємодії з нижнім контейнером. Крім того, якщо зловмисник виявить, що за ним стежить приманка, цілком ймовірно, що він може почати атаку на платформу, на якій вона розміщена. Як зазначив Упі Таммінен, розробник приманки Kippo: «Запустивши Kippo, ви практично атакуєте нападників. Точно так само, як і в реальному житті, роблячи щось подібне, ви краще знаєте, як захищатись!» [69].

Загальні запобіжні заходи, визначені щодо обмеження можливостей контейнера, включають:

- Запуск контейнерів приманок від імені користувача без повноважень *root*, обмеження рівня привілеїв для контейнерів по замовчуванню.
- Встановлення мінімальної кількості утиліт Linux, щоб у розпорядженні зловмисника, якому вдасться взаємодіяти безпосередньо з контейнером, було дуже мало інструментів.

Окрім цього, в якості частини конструкції мережі honeynet потрібен компонент, який забезпечує функціональність шлюзу honeywall, як описано в розділі 1.2.3. Цей засіб є першим компонентом honeynet, з яким зловмисник буде взаємодіяти, і тому він повинен бути максимально привабливим як шлюзовий маршрутизатор, який в разі зламу обіцяє зловмисникові можливість скомпрометувати багато інших пристроїв.

Щоб розробити ефективне рішення з використанням приманки, остаточний проект повинен включати заходи для задоволення ряду таких вимог:

- Повинна бути передбачена простота у налаштуванні та підтримка управління кожним компонентом honeynet незалежно.
- Приманки в мережі honeynet повинні бути максимально ізольовані від хост-системи.
- Масштабування повинно бути простим і «на льоту», додаючи або видаляючи приманки.
- Конфігурація honeynet повинна бути переносною, щоб його можна було просто видалити та повторно розгорнути.
- Для зловмисника, що взаємодіє з компонентом honeywall, повинна бути можливість бачити конфігурацію решти приманок в honeynet, полегшуючи поширення атаки та збір цінних даних.

Було добре відомо, що розгортання і налаштування мережі приманок, яка задовільняла б ці вимоги, є складним, довготривалим і схильним до помилок процесом, особливо з урахуванням обмеженого попереднього досвіду роботи з мережевим стеком Linux.

З огляду на те, що вже було вирішено розмістити проект на примірнику AWS EC2, який надалі називається «Honeypot», його необхідно виділити для розміщення декількох контейнерів, об'єднаних в мережу, щоб вони виглядали як окремі машини всередині тієї самої мережі. Як обговорювалось в статті Ланса Шпіцнера [38], honeywall, безсумнівно, є одним з основних компонентів до реалізації honeynet: він повинен бути здатний використовуватися як виділений пристрій в мережі, який використовується для управління пристроями в окремій зоні безпеки (Jumpbox), з можливостями мережевого підключення, щоб дати зловмисникові можливість атакувати доступ до іншої частини мережі.

Як зазначено в розділі 1.2.1, для того, щоб система на основі приманок була придатною для використання у виробничому середовищі, важливо, щоб її можна було легко розгортати та обслуговувати. Зокрема, якщо приманка була скомпрометована, її потрібно просто видалити та повторно розгорнути в

системі, зберігаючи при цьому усі дані, які були захоплені. Контейнери були визначені як засіб, який забезпечує велику кількість з цих необхідних функцій.

Однією із значних переваг контейнерних технологій є простота розгортання ідентично налаштованих середовищ. Однак та ж легкість розгортання для повно-мережевої системи за допомогою контейнерних технологій досягнута бути не може. Тому важливо, аби механізм розгортання для всієї системи розроблявся таким чином, щоб його можна було видалити та відновити за мінімальний проміжок часу.

Підхід до досягнення цієї мети полягав в тому, щоб зафіксувати конфігурацію системи у вигляді набору сценаріїв *bash*, які можна використовувати для автоматизації розгортання всього середовища honeynet. Ці сценарії можна підтримувати й оновлювати у міру розвитку конфігурації системи, тому, якщо система буде видалена, вона може бути ідентично розгорнута знову. Також вони повинні включати деякі прості заходи зручності використання, щоб допомогти в ефективному і простому розгортанні системи.

2.1.2 Моніторинг інцидентів

Компонент моніторингу інцидентів у запропонованій системі буде управляти агрегацією, обробкою та візуалізацією даних, а також генерувати попередження про виявлення загроз.

Цілком очевидно, що система моніторингу, на яку буде покладатися адміністратор для отримання ключової інформації про загрози, мінімально схильна до ризику. Цей фактор робиться більш важливим з огляду на те, що пропонована система, яка керується приманкою, насправді призначена для заохочення атак.

На цій підставі було вирішено, що фізична ізоляція системи моніторингу від системи Honeypot в максимально можливій мірі зможе знизити ймовірність виникнення небезпечних подій. Деяку ізоляцію забезпечить розміщення

системи моніторингу на незалежному віддаленому примірнику EC2 (надалі цей компонент системи буде називатись примірником управління). Однак виникають деякі питання стосовно того, як приховати зв'язок між примірником Honeypot і системою віддаленого моніторингу, оскільки зловмисникові небажано помічати регулярний зв'язок між однією зі своїх жертв і іншим цінним пристроєм.

Очевидно, що у зловмисника не повинно бути можливості переглядати або перенаправляти журнали, які створюються приманкою, в інше місце призначення, коли вони транспортуються між приманкою і системою моніторингу. Таким чином, механізм для передачі журналів між примірниками повинен підтримувати як аутентифікацію джерела/призначення, так і конфіденційність і цілісність. Для задоволення цих потреб буде використовуватись інструмент Filebeat.

Filebeat – це агент доставлення файлів журналів з відкритим вихідним кодом, який переносить їх з клієнтських серверів на інший хост. Важливо відзначити, що за допомогою Filebeat можна додати необхідний рівень безпеки, створивши сертифікат SSL і пару ключів для сервера управління. Згодом сертифікат можна використовувати спільно з екземпляром Honeypot, гарантуючи, що Filebeat відправляє зашифровані дані тільки на довірений сервер управління і навпаки [70].

Ключовим компонентом пропонованого монітора кіберінцидентів, безсумнівно, є візуалізація, яка покликана підвищити зручність використання та інформаційну цінність шляхом короткого опису даних про атаки, зібрані приманками.

Добре зарекомендованим засобом аналізу, агрегування і візуалізації журналів є стек ELK. Рішення про його використання було прийняте на підставі успішного використання стеку ELK в розробці TPot [50] і того факту, що він схвалений як підхід до візуалізації для приманки Cowrie [53].

Останнім компонентом системи моніторингу є механізм оповіщення про вторгнення. Найкращим рішенням для запропонованої системи є PSAD – інструмент виявлення вторгнень з відкритим вихідним кодом, який використовується для виявлення сканування портів та іншого шкідливого трафіку в системах Linux. PSAD працює шляхом моніторингу мережевих журналів пристрою і на підставі цього визначає, чи відбулося подія сканування або атаки.

До цього, як потенційний варіант розглядався інший механізм оповіщення від розробників стеку ELK і Filebeat – пагін XPack, який має здатність генерувати повідомлення на основі журналів, отриманих стеком ELK. [71] Однак після детального розгляду з'ясувалося, що попередження відправляється адміністратору системи тільки після того, як журнали приманки були оброблені екземпляром управління, тобто існує потенційна ймовірність необмеженої затримки в часі до вручення повідомлення. Це мотивувало рішення використовувати PSAD, який буде генерувати повідомлення, як тільки буде виявлена підозріла активність.

2.2 Проблеми щодо реалізації

На додаток до міркувань функціонального проектування і рішень, викладених в попередньому розділі, при реалізації пропонованої системи виник ряд специфічних проблем. Як частина проекту системи, вони були ретельно розглянуті та вирішені.

2.2.1 Зняття цифрових відбитків в середовищі приманок

Як обговорювалось в розділі 1.2.3, зняття цифрових відбитків системи зловмисниками в середовищах приманок являє собою постійну проблему для розробників і дослідників. У своєму дослідженні поведінки зловмисника з

використанням приманки Cowrie, Т. Баррон та ін. підкреслюють ймовірність того, що інформований зловмисник без завад може виявити середовище як несправжнє і відключитися [49]. Наприклад, за замовчуванням Cowrie привласнює ім'я хоста svr04, тому зловмисник, якому відомі властивості Cowrie, без проблем дізнається, що він взаємодіє з приманкою.

Однак, щоб провести справедливе порівняння між впливом різних конфігурацій приманок в експериментах цього дослідження, в якості контролю було важливо мати просту, готову до встановлення в мережу honeynet приманку Cowrie. Таким чином, був зроблений висновок, що, попри проблеми зі зняттям відбитків системи, така приманка повинна бути в кожній ітерації експерименту.

2.2.2 Етичні проблеми приманок

Ряд потенційних етичних проблем, пов'язаних з використанням приманок, вже були описані в розділі 1.2.3. Ще однією проблемою, яка потребує розгляду, є можливість скомпрометувати приманку, а потім використовувати її як платформу для атак на інші пристрої.

Досить важко збалансувати отримання корисних результатів досліджень з притаманними ризиками впровадження. Це міркування потребувало ретельного обдумування, і врешті решт були зроблені наступні висновки:

1) Ризик компрометації приманки можна сміливо розглядати як загальний ризик, пов'язаний з використанням пристроїв, підключених до мережі, і не обов'язково пов'язаний з використанням приманок зокрема. Ця думка збігається з думкою М. Навроцького та ін., які обговорювали цю проблему в опитуванні з програмного забезпечення приманок [37].

2) Хоча неможливо повністю виключити ризик поширення атаки, компоненти системи повинні бути реалізовані таким чином, щоб забезпечити його зниження. Контейнери для розміщення Cowrie будуть значною мірою

засновані на офіційному образі Docker-Cowrie Docker [55], який створено прихильниками спільноти розробників Cowrie. Цей образ контейнера запускає приманку під обліковим записом користувача без повноважень root з мінімальною кількістю доступних служб, тому навіть якщо зловмисник обійшов додаток Cowrie й отримав прямий доступ до контейнера, малоймовірно, що він зможе запустити будь-які атаки на інші пристрої.

3) Спеціалізовані дослідницькі середовища, в яких приманки ізольовані від важливих пристроїв і мереж, також ефективні для захисту зовнішніх систем.

2.2.3 Збереження вмісту енергозалежного контейнера

Характерною рисою контейнерів Docker є те, що за замовчуванням їх вміст не зберігається після того, як вони перестають працювати: як пояснили розробники проекту TPot, «всі дані в Docker є енергозалежними. Як тільки відбувається збій контейнера, всі дані, створені в його середовищі, зникають, а свіжий примірник перезапускається» [64]. Крім можливості запуску контейнера без прив'язки до конкретного хосту, відокремлюючи файлову систему контейнера від файлової системи його хоста, можна домогтися підвищеної безпеки шляхом ізоляції.

Збереження журналів, які генеруються приманками усередині цих контейнерів, має вирішальне значення. На думку розробників TPot, розв’язання проблеми нестабільності даних полягає в тому, щоб мати «постійне сховище... на хості, щоб зробити (дані) доступними та постійними при перезапуску контейнера або системи». [64] Це може бути досягнуто шляхом використання томів Docker, механізмів збереження, доступних для каталогів контейнерів. Томи Docker – це рекомендований підхід до збереження даних, які генеруються і використовуються контейнерами Docker. Коли контейнер звертається до томів, драйвер сховища Docker обходиться, і контейнер безпосередньо взаємодіє з підключеним томом у файловій системі хоста.

З урахуванням цих міркувань було вирішено, що в дослідницькому середовищі для збереження даних журналів, що генеруються приманками, слід використовувати саме томи Docker.

2.2.4. Міркування щодо безпеки використання Doker

Хоча команда розробників Docker історично відмінно справляються із завданням забезпечення безпеки контейнерів і надання широкої документації, [72] існують ризики, пов'язані з використанням їх ПЗ, які були виділені в літературі [46];[68];[62];[73];[61]. Під час здійснення проекту ці ризики розглядались дуже ретельно.

Найбільша проблема безпеки, яку було виявлено щодо використання контейнерів Docker, полягає в тому, що зловмисник може поширити атаку на систему, в якій ці контейнери розміщені. Компрометація цієї системи вкрай небажана.

Контейнери Docker спільно використовують ядро хоста, і багато дослідників стверджую, що це збільшує поверхню атаки в порівнянні з віртуальними машинами. Як описано Дж. Челладурай та ін., «віртуальні машини можуть взаємодіяти тільки з ядром віртуальної машини, а не з хостом» [73]. Й. Кедрович та П. Писарчик своїх статтях про використання контейнерів для розгортання приманок також розділяються цю стурбованість [62];[61].

Як описано в розділі 2.2.3**,** кілька томів Docker використовуються для підключення каталогів в хост-системі до кожного контейнера. Це створює потенційну загрозу безпеці, оскільки зловмисник всередині контейнера приманки потенційно може безпосередньо взаємодіяти з томом пов'язаного хоста і поміщати в зіставлений каталог все, що він хоче.

Відомо, що механізм Docker автоматично створює кілька віртуальних мереж на хості, на якому він працює, щоб контейнери могли взаємодіяти з хостом і один з одним. Будь-яке з'єднання між контейнерами та їх хостом є

ризиком для хоста, оскільки це може привести до того, що зловмисник під’єднається до хост-машини через мережу Docker і скомпрометує реальний сервер.

Ці ризики не є несуттєвими, тому для їх правильного розгляду було необхідно провести ретельний аналіз. Результатом є ряд розумних запобіжних заходів, які повинні бути включені в початковий проект, а саме:

- Характеристика ізоляції, описана в розділі 1.3.2 щодо контейнерів, враховує проблеми, пов'язані зі збільшенням поверхні атаки контейнерів в порівнянні з віртуальними машинами. Оскільки кожен контейнер працює у своєму власному просторі імен і використовує свій власний мережевий стек, за умови, що увага приділяється загальним ресурсам, до яких він має доступ, і рівню привілеїв, виділених для контейнера, його можна досить добре ізолювати від хост машини. Докладно це обговорюється в документації з безпеки Docker [72].
- Всі контейнери приманки, які будуть використовуватися в дослідженні, повинні бути налаштовані для запуску мінімальної кількості сервісів, які ймовірно можуть полегшити атаку. Це ще одна міра, призначена для забезпечення того, щоб дії зловмисника всередині контейнера були відносно обмеженими.
- Контейнери, які будуть використовуватися при реалізації honeynet, повинні бути побудовані на добре підтримуваних базових образах ОС з офіційних репозиторіїв Docker Hub [74]. Це гарантує, що останні оновлення безпеки, доступні для базових образів, успадковуються всіма контейнерами як надійна основа безпеки.
- Існує ефективний захід, визначений Дж. Челладхурай [73], який полягає в тому, щоб максимально обмежити мережеві можливості, забезпечуючи зв'язок тільки між довіреними контейнерами. Щоб реалізувати це, необхідно приділити особливу увагу конфігурації контейнерів, аби вони

не відкривали лишні порти і не належали до будь-яких непотрібних мереж.

2.2.5. Привабливість приманок для заохочення атак IoT ботнетів

Очевидно, що для привернення уваги ботнетів IoT, характеристики приманок повинні бути якомога більш привабливими [20];[23];[54]. У своїй статті розробники IoTPot обговорюють важливість підтримки всіх варіантів, які хочуть використовувати зловмисники, забезпечення реалістичних інтерфейсів входу і полегшення входу в систему при проектуванні успішної IoT приманки [58].

Після аналізу відповідних публікацій і досліджень, ряд характеристик пристроїв IoT, які можуть залучати ботнети IoT, був визначені в такий спосіб:

- Наявність на пристрої протоколів SSH і/або telnet.
- Застаріла або вразлива версія служб, наявних на пристрої або в ОС.
- Наявність «цікавого» мережевого трафіку до пристрою IoT або від нього, особливо якщо цей трафік не зашифрований, (аналогічно тому, що було реалізовано Даулінгом та ін.) [14].
- Можливість ідентифікувати пристрій як продукт певного виробника, пристрої якого широко відомі як вразливі.

Всі ці характеристики можна розглядати як потенційні змінні в майбутніх експериментах, які допоможуть спроектувати ефективну адаптивну приманку, орієнтовану на бот-мережі IoT.

2.2.6 Виявлення атак IoT ботнетів

Ще одна складність в оцінці будь-яких результатів, отриманих мережею honeynet, полягає в тому, яким чином можна відрізнити атаки зловмисників від автоматичних атак ботів. Ця різниця особливо важлива для другої мети

дослідження, яка полягає в розробці ефективні приманки, призначеної для атак бот-мереж IoT.

У дослідженні, яке деталізує спостереження про поведінку зловмисників, Т. Баррон та ін. звернули увагу, що «боти виконують особливі дії, які не залежать від середовища... на людей, які зазнали атаки, впливає базове середовище, наприклад, виконання більшої кількості команд на приманки з реалістичними структурами файлів і тек» [49]. Деякі з їх більш конкретних висновків:

- Люди звертають увагу на користувацькі файли з реалістичними назвами та наповненням, а боти – ні;
- Люди схильні до орфографічних помилок при виконанні консольних команд, а боти – ні;
- Боти будуть поводитись однаково незалежно від того, на якому хості вони працюють (тобто не залежать від середовища);
- Послідовність команд атаки ботів є автоматизованою, що означає мінімальну затримку в часі між введенням цих команд;
- Більшість атакуючих є ботами.

Щоб відрізнити бот-мережі IoT від інших бот-мереж, найбільшу цінність становлять зауваження з літератури, яка стосується активних бот-мереж IoT [20];[54]. Наприклад, IoT ботнети Mirai і Hajime, як відомо, перевіряють наявність оболонки BusyBox на пристроях жертви [75]. Цей тест проводиться для того, щоб переконатися, що на пристрої встановлено справжня оболонка Linux. Як пояснено С. Едвардсом та ін. в статті, присвяченій ботнету Hajime IoT [54], «приватний CLI, швидше за все, відхилить команду, але легітимна оболонка Linux виконає Busybox... повідомивши Hajime, що у нього є справжня оболонка Linux». Ці види індикаторів, які можна спостерігати при вивченні зареєстрованих сеансів атак, також допоможуть визначити тип ботнету за будь-якими атаками, які захоплено приманкою.

2.2.7 Достовірність оцінки експериментів з приманкою

Як вже обговорювалось в розділі 1.2.2, використання приманки в пропонованому дослідницькому середовищі дозволяє достовірно оцінити відносну ефективність їх різних характеристик. Проектування такої експериментальної системи, однак, вимагає, щоб між приманками було якомога менше змінних. Таким чином, в приманку необхідно включити:

- простий базовий «керуючий» пристрій без поліпшень;
- Безліч поліпшених приманок, кожна з яких демонструє різні варіації однієї й тієї ж потенційно привабливої характеристики.

Використання цього підходу означає, що для цілей оцінки джерела мінливості в експерименті максимально контролюються: всі приманки будуть піддаватися атаці в однакових умовах середовища з однією контрольованою зміною на приманку. Такий підхід усуває необхідність враховувати багато істотних змінних середовища, наприклад, якби кожна приманка оцінювалась незалежно на різних хост-серверах. В цьому випадку такі змінні можуть поставити під сумнів достовірність тверджень про відносну ефективність різних характеристик приманки.

2.3 Висновок до розділу до проектування

З самого початку було визнано, що з урахуванням непередбачуваності досліджень і, зокрема, непередбачуваності атак, вимоги до системи та кінцевий проект, ймовірно, будуть розвиватися в міру його просування. Проте, міцний, надійний і продуманий дизайн, який засновано на обґрунтованих рішеннях і який підтримується попередніми дослідженнями та поточною передовою практикою, безсумнівно, внесе позитивний внесок в ефективне впровадження спільного рішення.

В рамках оновлення результатів досліджень для цієї системи, щоб дати загальний огляд архітектури та операцій системи. була підготовлена концептуальна схема, яка представлена на рисунку 2.1:

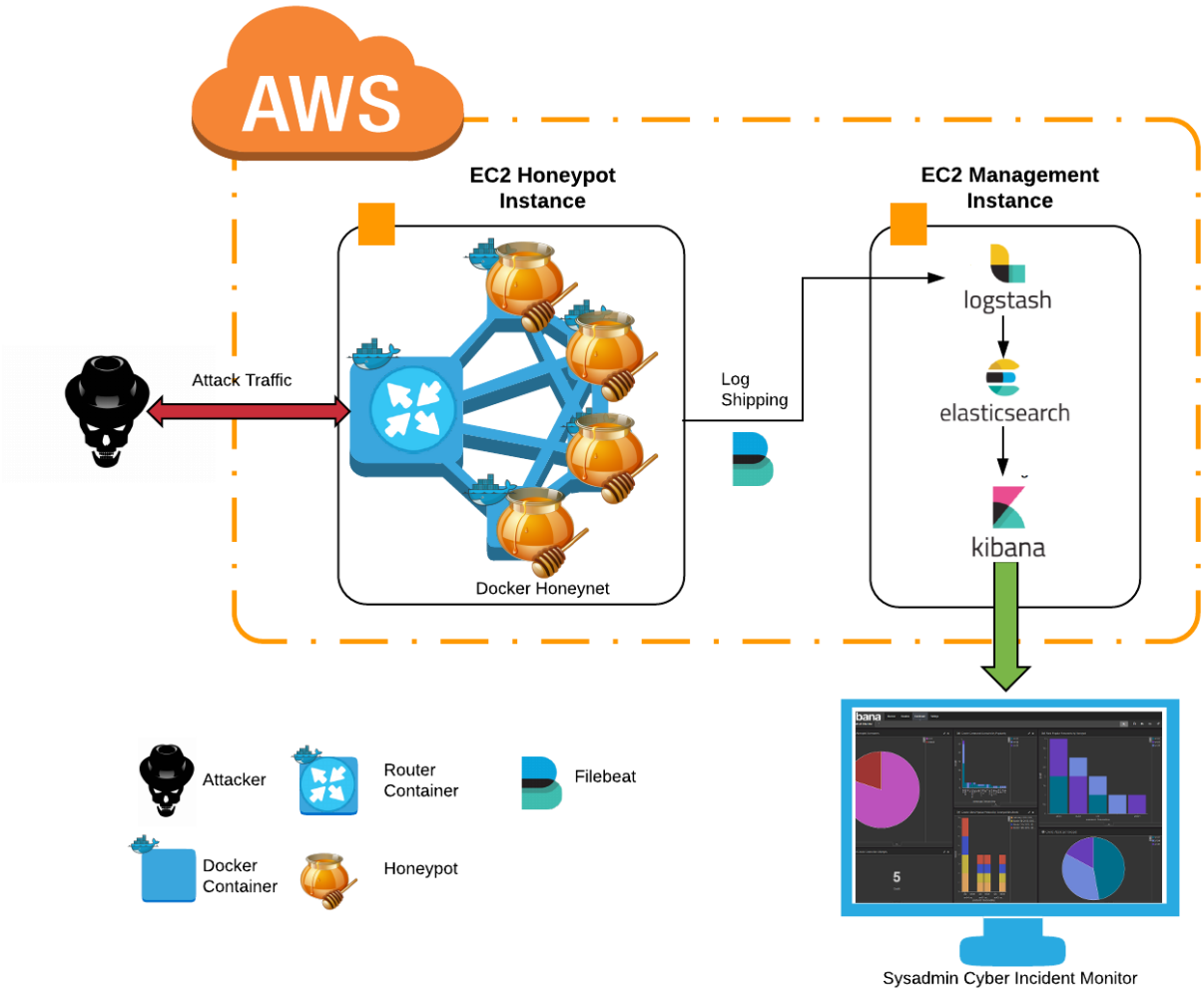

Рисунок 2.1 – Огляд пропонованої конструкції системи

Компоненти нараховують: 2 сервера AWS EC2; контейнерну мережу, яка складається з контейнерних honeypots, розміщених в примірнику Honeypot EC2; систему обробки і візуалізації журналів ELK, яка розміщена в примірнику управління EC2; доступ через веб-консоль до візуалізацій, що генерується системою.

3 РЕАЛІЗАЦІЯ ДОСЛІДНИЦЬКОГО СЕРЕДОВИЩА HONEYNET

3.1 Налаштування контейнерних приманок

Для того, зрозуміти можливості AWS EC2, приманки Cowrie і платформи Docker, деякі початкові дослідження того, як ці компоненти працюють на практиці, послужили корисною відправною точкою. Спочатку було розгорнуто примірник AWS EC2, далі послідувало повне встановлення приманки Cowrie і, нарешті, були проведені деякі експерименти з Docker і його компонентами.

3.1.1 Розгортання примірника AWS EC2

Один примірник Amazon EC2 був отриманий в рамках пробного користування – пропозиції, що дозволяє користуватись сервісам AWS певний період. Ця пропозиція включає 750 годин обчислювального часу з одним віртуальним ЦП і 2 ГБ оперативної пам'яті на термін до 12 місяців.

Підготовка примірника EC2 - це простий і послідовний процес, який охоплює створення облікового запису AWS і слідування майстру налаштувань розгортання. В цьому випадку примірник сервера Ubuntu 16.04 LTS був підготовлений з максимальним розподілом ресурсів, яким обмежувалась пробна версія.

В рамках розгортання примірника EC2 було згенеровано відкритий ключ SSH (шифрування, що використовується AWS для генерації цих ключів, являє собою 2048-бітове шифрування SSH-2 RSA). Цей ключ можна використовувати для безпечної аутентифікації з екземпляром по SSH замість автентифікації по паролю.

Для управління вхідним і вихідним трафіком в AWS EC2 використовуються групи безпеки. Вони діють як віртуальний брандмауер з набором правил безпеки, які визначають діапазони адрес, протоколи та порти, дозволені для зв'язку з примірником. Група безпеки EC2 за замовчуванням дозволяє доступ SSH через порт 22/TCP з усіх адрес IPv4 (рисунок 3.1)**.** Варто

зазначити, що з кожним примірником EC2 повинна бути пов'язана хоча б одна група безпеки.

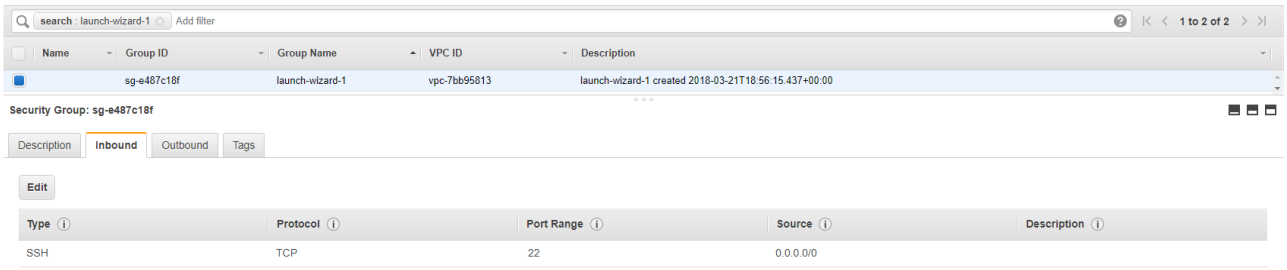

Рисунок 3.1 – Правила безпеки за замовчуванням AWS EC2

3.1.2 Налаштування приманки Cowrie

Перш ніж намагатися інкапсулювати приманку Cowrie в контейнерне середовище, необхідно зрозуміти її базову конфігурацію і налаштування. Це і є наступним кроком в розумінні деталей того, як пропонована система буде реалізована.

Як обговорювалося в розділі 1.4.1, Cowrie - це приманка з середнім рівнем взаємодії, яка призначена для реєстрації атак і взаємодій з оболонкою. Докладний посібник з встановлення приманки розміщено на Github [76]. Слідуючи керівництву, приманку було відносно просто налаштувати. Це пояснюється тим фактом, що Cowrie працює як Python virtualenv - інструмент для створення ізольованих середовищ Python [77]. На цьому етапі важливою вимогою було забезпечити, щоб Cowrie запускався користувачем без повноважень *root*, оскільки зловмиснику має бути важко домогтися підвищення привілеїв, якщо йому вдасться взаємодіяти безпосередньо з хостом. В цьому відношенні Cowrie забезпечує мережу безпеки - якщо хтось спробує запустити приманку від імені користувача root, він отримає повідомлення «Помилка! Ви не повинні запускати Cowrie від імені root!».

Найзначнішим етапом в процесі налаштування було включення Cowrie для отримання трафіку атаки, призначеного для примірника EC2. За

замовчуванням приманка прослуховує вхідний трафік через порти 2222/TCP і 2223/TCP хоста і, таким чином, для отримання трафіку від привілейованих портів 22/TCP (SSH) і 23/TCP (telnet) деякий рівень переадресації трафіку був обов'язковим. Для досягнення цього в керівництві по встановленні Cowrie пропонуються два підходи: [76]

1) Переадресація портів

Переадресація портів включає використання правил *iptables* для пересилання вхідного трафіку через порти 22/TCP (SSH) і 23/TCP (telnet) на порти 2222/TCP і 2223/TCP.

2) Пряме прослуховування з використанням Authbind

В якості привілейованих портів для прослуховування вхідного трафіку на 22/TCP і 23/TCP потрібні привілеї користувача root. Cowrie не працюватиме з такими привілеями, тому можна використовувати утиліту authbind Linux, яка дозволяє йому безпосередньо прослуховувати трафік на цих портах. Саме цей підхід було використано надалі.

Останнім кроком перед налаштуванням Cowrie було оновлення конфігурації SSH на примірнику хоста, щоб адміністратор як і раніше мав доступ до нього ззовні. Це необхідно, оскільки весь вхідний трафік через стандартний порт SSH 22/TCP тепер перенаправляється в Cowrie і вимагає простого редагування файлу конфігурації /etc/ssh/sshd, змінюючи значення порту з 22/TCP на інший доступний номер.

Загалом, існує три основних компоненти Cowrie, які можна використовувати для налаштування середовища приманки:

- *userdb.txt*

Цей файл використовується для визначення комбінацій імені користувача і пароля, які приманка повинна прийняти від зловмисника. Комбінації за замовчуванням зображені на рисунку 3.2. Перший запис *root:x:!root* вказує, що ім'я користувача *root* має бути прийнято, але не тоді, коли зазначений пароль також *root*. Третій запис *root:x:** вказує, що ім'я користувача *root* має бути прийнято з будь-якими іншими паролями, для яких не вказано жодних правил.

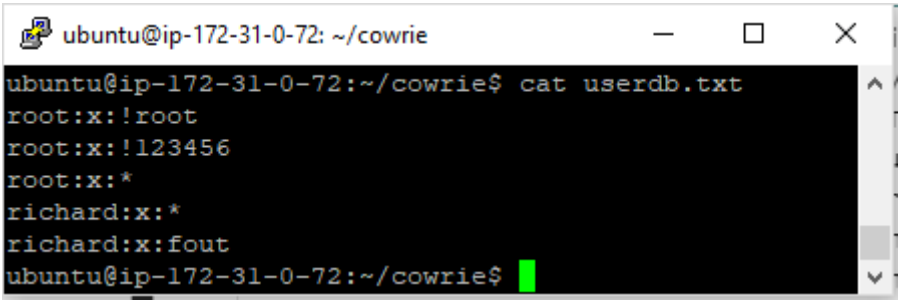

Рисунок 3.2 – Паролі приманки Cowrie за замовчуванням

- *cowrie.cfg*

Цей файл є основним файлом конфігурації, який використовується Cowrie для визначення того, як налаштувати середовище приманки. Він має безліч налаштувань, в тому числі для інтеграції з зовнішніми інструментами, а також інші параметри для управління тим, як середовище приманки буде виглядати для зловмисника.

- *honeyfs/*

Шляхом додавання файлів і каталогів в honeyfs/ можна налаштувати вміст файлової системи середовища приманки.

Для даного дослідження конфігурація всіх описаних вище компонентів була майже повністю змінена зі значення за замовчуванням.

Щоб отримати дані логування, приманка була налаштована і відкрита для публічного Інтернету. Cowrie був налаштований з ім'ям, яке відповідає конкретній моделі IP-камери TPLink, а саме TPLink-TL-SC3171, оскільки після проведення короткого веб-пошуку ця модель була відразу ж ідентифікована як з вразливістю віддаленого виконання коду (RCE). Група безпеки EC2 за замовчуванням залишилася незмінною, що означало, що приманка залишалась відкритою для всього адресного простору IPv4 на порті 22/TCP (SSH).

Середовище працювало протягом однієї години, і за цей час було зареєстровано атаки методом грубої сили з 4 різних IP-адрес. Згодом ці атаки були ідентифіковані як атаки ботів, вихідні IP-адреси яких визначені з різних країн, включаючи Єгипет, Росію, Китай і В'єтнам. Місцезнаходження джерел

було визначено за допомогою Maxmind GeoIP Database, яка приблизно визначає географічне положення IP-адрес [79].

Один з сеансів атаки викликав особливий інтерес, оскільки порівнюючи спроби входу в систему було виявлено, що деякі з паролів є такими ж, які були визначені в дослідженнях вихідного коду ботнету Mirai [20]. Ця знахідка добре ілюструє масштаб зараження пристроїв цим ботнетом і його варіантами.

Журнал Cowrie, в якому були записані дані сеансу атаки в форматі JSON, дав уявлення про рівень деталізації інформації, яку приманка може захопити. Інформація, отримана на цих сесіях, включала:

- IP-адрес джерела і порт з'єднання;
- Версію протоколу віддаленого SSH і алгоритм шифрування;
- Комбінації імені користувача і пароля, які були використані, а також те, чи була спроба успішною або невдалою;
- Інформація про те, як довго підтримувалося з'єднання і причина, по якій воно було перервано;
- Мітки часу, відповідні кожній події, і унікальний ідентифікатор сеансу для кожного з'єднання.

3.1.3 Побудова контейнерів Docker

Екосистема Docker є досить простою, але складається з ряду окремих компонентів, які разом дозволяють розміщувати додатки в контейнерах без збереження стану. Розуміння компонентів цієї екосистеми і того, як вони взаємодіють, є важливим для ефективного використання цієї технології та реалізації мережі Honeynet.

Щоб зрозуміти принцип побудови контейнерних середовищ і того, як з ними взаємодіяти, перш ніж перейти до реалізації контейнерної медової мережі, було розглянуто базовий вихідний код Docker-Cowrie, наданий розробниками Cowrie [55].

Середовище контейнера Docker повністю визначається його образом, який може бути наповнений за допомогою Dockerfile. Dockerfile складається з

набору текстових команд, які використовуються для побудови образу і визначає, як виглядатиме середовище всередині контейнера. Всі зовнішні файли, на які посилається Dockerfile, є частиною образу.

Життєвий цикл контейнера, який часто називають циклом “складання і запуску” (англ. – «build-and-run»), показаний на рисунку 3.3:

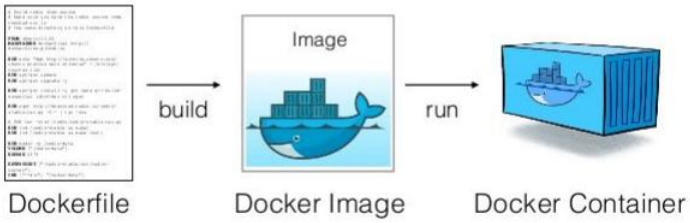

Рисунок 3.3 – «Bulid-and-run» цикл Docker

Dockerfile використовується для створення образу Docker, який, у свою чергу, використовується для запуску контейнера.

1) Створення Docker-контейнера

Образ контейнера генерується виконанням CLI команди *docker build* з посиланням на Dockerfile. Інструкції в Dockerfile виконуються одна за одною і поступово додаються в новий образ, перш ніж, нарешті, генерується ідентифікатор. Новий образ зазвичай використовує базовий образ як основу.

2) Запуск Docker-контейнера

Якщо CLI команда *docker create [arguments] [image-ID]* виконується з посиланням на ідентифікатор образу, то на його основі створюється доступний для запису рівень контейнера, який готовий до роботи з використанням будь-яких наданих аргументів. Проте, контейнер не працює автоматично після створення: для запуску використовується CLI команда *docker start [container-ID]*.

3) Зупинка або видалення контейнера

Зупинити контейнер можна за допомогою команди *docker stop [container-ID]*. Його також можна видалити, скориставшись командою *docker container rm*

[container-ID], але тільки в тому випадку, якщо він не запущений, або запущений, але видалення викликане із зазначенням додаткових аргументів.

Різні елементи середовища контейнера повинні бути налаштовані в певних точках його життєвого циклу: деякі параметри конфігурації можуть бути вказані до виконання, тоді як інші повинні бути вказані вже під час. Наприклад, внутрішню таблицю маршрутизації контейнера неможливо оновити до тих пір, поки цей контейнер не буде запущений, і, отже, ця конфігурація не може бути вказана безпосередньо як частина базового образу Docker. Це здається перешкодою, оскільки означає, що не всі конфігурації середовища можуть бути безпосередньо визначені в образі контейнера. Тут стає корисним використання команди Docker *ENTRYPOINT*.

Команда *ENTRYPOINT* може використовуватися в Dockerfile для вказівки сценарію, який повинен запускатися разом з образом контейнера. Це означає, що сценарій, який визначає поновлення внутрішньої таблиці маршрутизації контейнера, можна вказати як команду *ENTRYPOINT*. Після запуску контейнера цей скрипт буде виконаний і таблиця маршрутизації буде оновлена. Це дуже корисна і гнучка опція в конфігурації контейнерного середовища.

Як засіб збереження мінливого вмісту, що генерується контейнером в Docker використовуються томи. Вони можуть бути змонтовані за допомогою команди *VOLUME* у файлі Dockerfile для образу контейнера, а відповідний каталог хоста, до якого вони зіставлені, задається як аргумент для *CLI* команд *Docker create* або *docker run* на більш пізньому етапі. Після зупинки контейнера всі томи, підключені до нього, і їх вміст зберігаються.

На рисунку 3.4 показано концептуальне уявлення того, як можна використовувати томи для збереження вмісту, згенерованого в контейнері.
Файли в цих томах в певному каталозі кожної системи доступні як з хоста, так і з контейнера. Зміни в вмісті будь-якого каталогу можуть бути внесені динамічно, і вони відразу ж будуть доступні у відповідному відображеному каталозі.

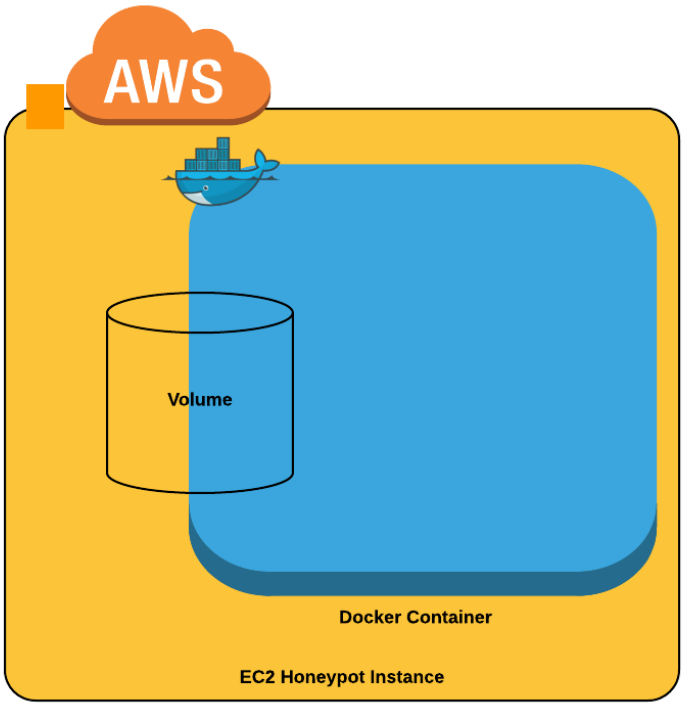

Рисунок 3.4 – Концептуальне уявлення томів Docker

Контейнери, як і фізичні машини, мають власний мережевий стек. Кожен контейнер має свої власні порти та віртуальні інтерфейси, що з'єднують їх з віртуальними мережами Docker. Це ключовий елемент функціональності, який робить Docker відповідним для реалізації медової мережі в запропонованій системі.

Існує два способи надання портів – через CLI команди *docker create* і *docker run*.

- За допомогою аргументу *-expose*, який дозволяє відкрити вказаний порт контейнера для хост-комп'ютера та інших контейнерів. Однак він не публікується в мережеві інтерфейси хоста.
- З іншого боку, використання аргументу *-publish* дозволяє опублікувати зазначений порт контейнера в мережеві інтерфейси хоста, що означає, що він доступний через ці порти в Інтернеті.

Аргумент *-expose* також може бути вказаний в образі Dockerfile, що означає, що коли контейнер будується, зазначені порти надаються тільки для між контейнерної взаємодії [55].

Якщо в мережі не вказано додаткових умов, усі контейнери після створення за замовчуванням автоматично підключаються до мережі моста Docker. Ця мережа дозволяє контейнерам взаємодіяти безпосередньо один з одним і з хост-машиною через віртуальний мережевий інтерфейс, який називається *docker0*.

Розглянувши роботу AWS EC2, приманки Cowrie і контейнерної екосистеми Docker, було отримане тверде розуміння щодо наступних кроків у реалізації дослідницького середовища, описаного в розділі 4, і деяка базова конфігурація. Зокрема, на той час, коли всі з вищезгаданих функціональних можливостей Docker були вивчені, був розроблений елементарний сценарій розгортання *do_honeypot.sh*, призначений для швидкого розгортання і видалення контейнерів. Основними функціями сценарію є:

- Створення базового образу Docker-Cowrie;
- Створення контейнера на основі попередньо створеного образу з унікальним іменем;
- Запуск і зупинка контейнера із зазначенням його імені;
- Введення запущеного контейнера, представлення користувачеві внутрішнього інтерфейсу командного рядка (CLI);
- Надання інструкцій стосовно того, як використовувати сценарій для управління контейнерами в системі.

3.2 Розгортання хост-сервера

Для реалізації системи в зоні доступності Amazon US East (Огайо) під управлінням операційних систем Ubuntu 16.04 LTS було розгорнуто два примірники сервера AWS EC2: екземпляр сервера-приманки та екземпляр сервера управління. Кожному з них, в залежності від їх вимог до обладнання, були виділені різні апаратні ресурси. В обох випадках ресурсів, які надавались в межах 12-місячного пробного користування, було недостатньо, а отже, виникла потреба в дорожчих примірниках.

3.2.1 Примірник сервера приманки

Розміщення декількох контейнерів Docker в рамках пропонованої контейнерної медової мережі означало необхідність в досить потужному примірнику сервера. На основі первинного експерименту з запуском контейнерів Docker на пробному примірнику було підготовлено примірник EC2 з 16 ГБ оперативної пам'яті та 4 віртуальними процесорами.

Щоб привернути увагу IoT-ботів, які сканують Інтернет на наявність вразливих пристроїв, порти SSH (22/TCP) і telnet (23/TCP) повинні бути видні на цьому примірнику сервера за замовчуванням. В цілому, шляхом налаштування правил безпеки, тільки ці два порти й були доступні.

Також на примірнику був відкритий ще один порт, але з великими обмеженнями доступу. Як вже було описано в розділі 3.1.2, призначення цього порту – забезпечити збереження адміністративного доступу. Порт був обраний випадковим чином як 45678/TCP і відкритий тільки для IP-адреси примірника управління, щоб максимально приховати сам факт його відкритості.

Всі правила безпеки, налаштовані для примірника сервера-приманки, можна побачити на рисунку 3.5:

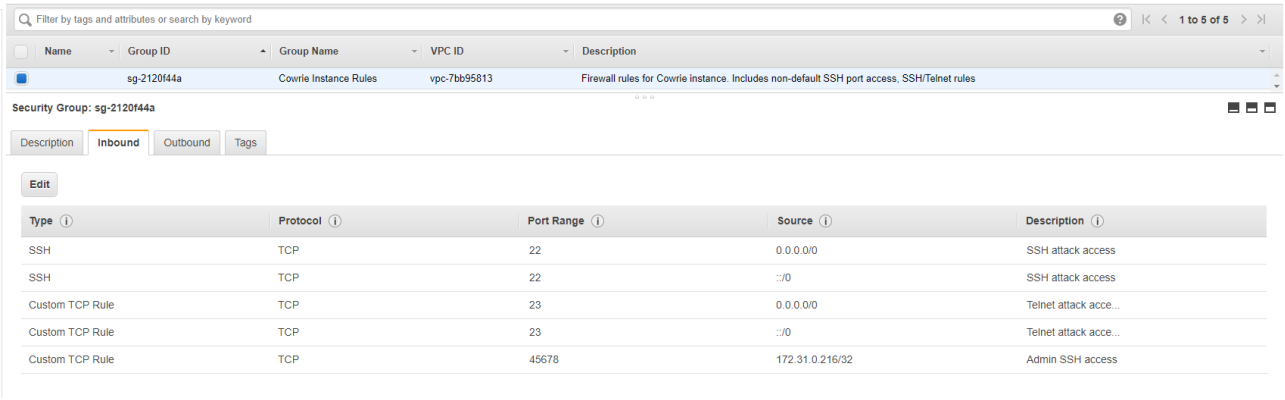

Рисунок 3.5 – Правила безпеки примірника сервера-приманки AWS EC2

Як можна бачити на рисунку, порт 80/TCP відкритий одній IP-адресі, з якої можна отримати доступ до згенерованих візуалізацій. Порт 5045/TCP дозволяє додатку Filebeat передавати зашифровані файли журналів на

примірник сервера управління. Порт 22/TCP також доступний для однієї IP-адреси, щоб дозволити адміністративний доступ до примірника.

3.2.2 Примірник сервера управління

Оскільки примірник сервера управління необхідний зокрема для розміщення стека обробки журналів ELK, для запуску цих інструментів були потрібні певні мінімальні апаратні ресурси. З міркувань вартості, для задоволення вимог було виділено мінімальний набір ресурсів: сервер з 4 ГБ ОЗУ і 2 віртуальними ЦП.

Загалом, на цьому примірнику було відкрито три порти:

- Порт 22/TCP для SSH доступу адміністратора;
- Порт 5045/TCP, який був обраний випадковим чином, щоб дозволити примірнику сервера-приманки передавати журнали примірнику сервера управління;
- Порт 80/TCP, щоб дозволити доступ HTTP через веб-браузер для перегляду візуалізацій.

Правила безпеки примірника можна побачити на рисунку 3.6:

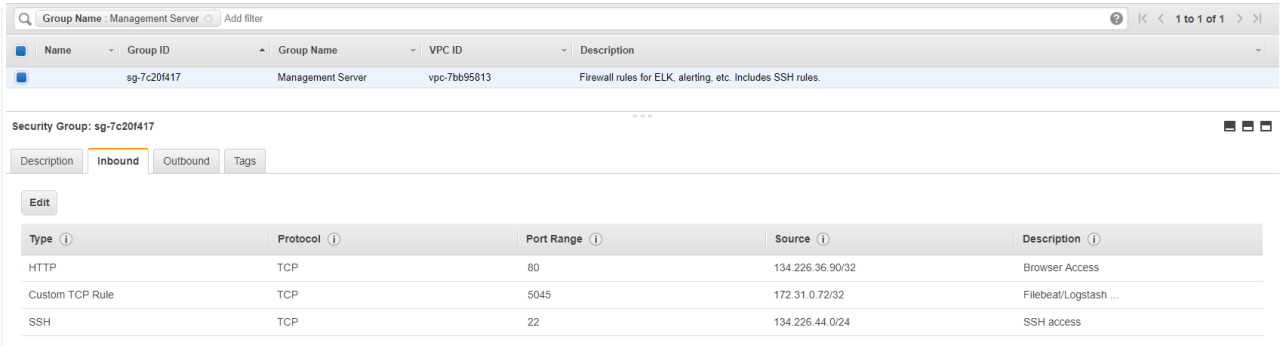

Рисунок 3.6 – Правила безпеки примірника сервера управління AWS EC2

Як проілюстровано на рисунку, порт 80/TCP відкритий одній IP-адресі, з якої можна отримати доступ до згенерованих візуалізацій через веб-сеанс. Порт 5045/TCP залежить від IP-адреси примірника сервера-приманки, і дозволяє додатку доставлення журналів Filebeat передавати зашифровані файли журналу

на примірник сервера управління. Порт 22/TCP також доступний для однієї IP-адреси, щоб дозволити адміністративний доступ.

3.3 Побудова мережі Honeynet в екосистемі Docker

Як вже зазначалось раніше, мережа Honeynet повинна розгортатися автоматично шляхом запису в сценаріях bash всіх команд, необхідних для побудови мережевого середовища. В ході розробки мережі сценарій розгортання *do_honeypot.sh* поступово розширювався.

3.3.1 Контейнеризація приманки Cowrie

На етапі проектування було вирішено, що образ Docker для контейнерів приманки Cowrie буде значною мірою заснований на існуючому образі, який розроблено та експериментально досліджено фахівцями проекту Cowrie [55]. Для цього є обґрунтовані причини:

- Образ базової ОС debian:jessie-slim запускається як основа для образу контейнера.
- Всі залежності (більшість з яких є залежностями збірки), необхідні для Cowrie, встановлені.
- Cowrie налаштовується відповідно до обов'язкових кроків встановлення, які зазначені в репозиторію Cowrie Github [76], перед видаленням всіх непотрібних пакетів збірки.
- Створюється користувач без повноважень *root* (і встановлюється як користувач за замовчуванням для контейнера).

Створення користувача без повноважень *root* за замовчуванням і видалення непотрібних пакетів збірки роблять середовище контейнера відповідним чином обмеженим в разі, якщо зловмиснику вдасться обійти додаток Cowrie і взаємодіяти безпосередньо з нижнім контейнером.

Cowrie здатний емулювати багато поширених утиліт Linux, включаючи *cat, wget, echo* та ін. Існує навіть можливість додавати нові емуляції утиліт,

якщо це необхідно. Однак, є дуже мало пакетів, які дійсно повинні бути встановлені в контейнері Cowrie, а ті, які потрібні, в основному вже включені в збірку [55].

В Dockfile Cowrie було налаштовано *entrypoint* сценарій для запуску під час виконання контейнера. Цей сценарій використовувався для налаштування *authbind*, щоб Cowrie міг прослуховувати трафік контейнера через порти 22/TCP (SSH) і 23/TCP (telnet). Його лістинг наведено нижче.

Лістинг 5.1 – Entypoint сценарій, зазначений в налаштованому Dockfile Cowrie

```
#!/bin/bash

# Дозволити cowrie слухати порти 22 і 23 як non-root
     touch /etc/authbind/byport/22 &&\
     touch /etc/authbind/byport/23 &&\
     chown cowrie:cowrie /etc/authbind/byport/22 &&\
     chown cowrie:cowrie /etc/authbind/byport/23 &&\
     chmod 777 /etc/authbind/byport/22 &&\
     chmod 777 /etc/authbind/byport/23

# Повідомити Cowrie що використовується authbind
sed -i 's/AUTHBIND_ENABLED=no/AUTHBIND_ENABLED=yes/'
     /cowrie/cowrie-git/bin/cowrie

## Запуск сервісу cowrie
su - cowrie -c ''/cowrie/cowrie-git/bin/cowrie start -n''
```

Лістинг показує налаштування *authbind* шляхом створення 2 нових порожніх файлів з використанням *touch*, і призначення права власності на ці файли користувачеві. Це дозволяє використовувати *authbind* для отримання вхідного трафіку через порти 22/TCP та 23/TCP. Потім приманка повинна бути повідомлена про використання *authbind*, що досягається налаштуванням параметра конфігурації *AUTHBIND ENABLED*. Нарешті, сценарій запускає приманку Cowrie під користувачем cowrie.

Як визначено в розділі 2.2.3**,** для динамічного обміну інформацією між контейнером і його хостом будуть використовуватись томи. В цій системі вони служать для двох цілей:

1. Доступність файлів конфігурації Cowrie, які можуть бути налаштовані на хості

2. Забезпечення доступності журналів, згенерованих в контейнері Cowrie, поза контейнером.

Таким чином, в сценарій розгортання мережі Honeynet були визначені функції для створення контейнерів Cowrie із зазначенням кількості томів, які повинні бути до них підключені. Загалом, щоб дозволити доступ до контейнера з хоста, були змонтовані наступні каталоги:

- Каталог *dl/*, в якому за замовчуванням зберігаються контрольні суми спроб завантаження зловмисниками.
- Каталог *log/*, в якому за замовчуванням зберігаються файли журналів, згенеровані Cowrie.
- Каталог *data/* в якому за замовчуванням розташований файл конфігурації пароля *userdb.txt*.

Через складнощі при монтуванні вкладених томів з хоста в контейнер, для копіювання файлу *cowrie.cfg* використовувалась команда *docker cp*.

Оскільки всі контейнери повинні мати можливість взаємодіяти один з одним через порти 22/TCP і 23/TCP, в Dockerfile Cowrie була використана команда *EXPOSE*. Її виконання означало, що при запуску порти 22/TCP і 23/TCP в контейнері Cowrie за замовчуванням будуть видимі будь-яким іншим контейнерам в мережі моста Docker.

У випадку контейнера Cowrie Honeywall ситуація дещо відрізнялася: щоб отримувати атаки, він повинен мати можливість доступу до вхідного трафіку через мережевий інтерфейс хоста. Для цього при створенні контейнера Honeywall використовувався аргумент *-publish* для зіставлення портів 22/TCP і 23/TCP на мережевому інтерфейсі хоста з відповідними портами на контейнері. Використання аргументу *-publish* також означало, що в мостовій мережі Docker контейнеру автоматично буде призначений віртуальний мережевий інтерфейс, щоб він міг отримувати вхідний мережевий трафік на ці порти від примірника хоста. Будь-який трафік, отриманий хостом через порти 22/TCP та 23/TCP,

відображається безпосередньо на відповідні порти контейнера, що робить їх доступними з Інтернету. Як це виглядає концептуально ілюструє Рисунок 3.7.

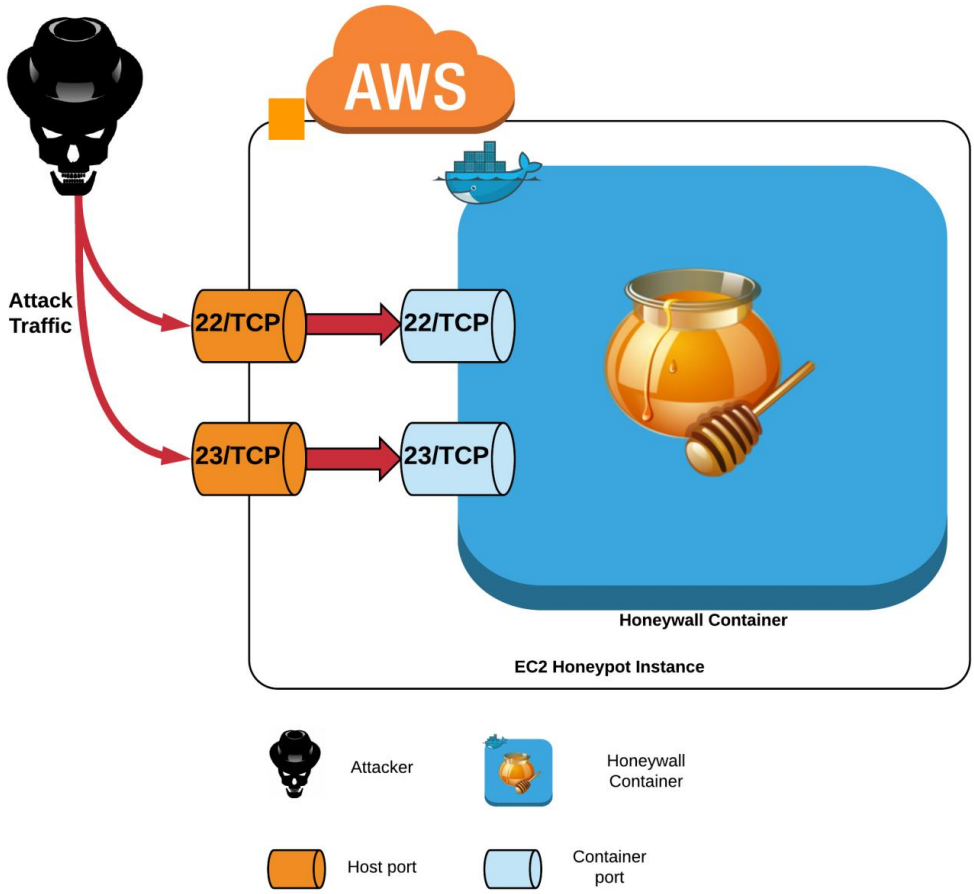

Рисунок 3.7 – Публікація порту в контейнері Honeywall

3.3.2 Розгортання мережі Honeynet

Docker мережа Honeynet є одним з нових компонентів конструкції системи й буде складатися з мережі Docker і декількох контейнерів Cowrie, які підключені до неї. Контейнер Cowrie Honeywall буде виконувати роль шлюзу між примірником сервера-приманки EC2 і цією мережею, а це означає, що йому будуть потрібні віртуальні інтерфейси як в мережі моста, так і в новій мережі.

Нова мережа Docker, в якій повинні розміщуватися контейнери Cowrie, була визначена як мостова мережа з адресою 10.0.0.0/820. Цю мережеву адресу було обрано тому, що вона зарезервована для використання в приватних

мережах, тобто тих, які не підключені безпосередньо до Інтернету [87]. Мережа отримала ім'я dmz, і відповідає поняттю DMZ, описаному в розділі 1.2.1.

На цьому етапі у сценарій розгортання *do_honeypot.sh* була додана функція, що дозволяє автоматично визначати мережу без необхідності надання відомостей про конфігурацію:

Лістинг 3.2 – Створення мережі dmz

```
# Local DMZ bridge network to which all containers are connected
Create_dmz_net( ) {
     # Define the network
     network_exists=$( docker network Is | grep “dmz” )
     if [[ -n “$network_exists” ]] ; then
          echo “DMZ network defined”
     else
          echo “Creating DMZ network (10.0.0.0/8)”
          docker network creat –d bridge \
          --subnet 10.0.0.0/8 \
          --attachable \
          dmz
     fi
}
```

Сценарій використовує команду *docker network ls*, щоб перевірити, чи існує мережа, і якщо ні – створює її. Аргумент *-d bridge* вказує, що для створення мережі повинен використовуватися мережевий драйвер моста. Аргумент *attachable* вказує, що контейнери можуть бути додані в мережу вручну «на льоту».

Як обговорювалося в розділі 2.2.4, при реалізації дослідницького середовища необхідно врахувати забезпечення того, щоб у контейнерів приманок не було ніяких лишніх мережевих можливостей. Якщо вказати в сценарій *do_honeypot.sh*, що контейнери Cowrie повинні бути підключені до мережі dmz, вони не будуть додані в інші мережі.

Підключення контейнерів до мережі dmz включало додавання в сценарій *do_honeypot.sh* одного додаткового аргументу: *-network «dmz»*. При виконанні команди *docker create* тепер випадковим чином буде призначатись IP-адрес в

мережі dmz для контейнерів Cowrie і певний IP-адрес для контейнера Honeywall.

Функції контейнера Honeywall відрізняються від функцій інших контейнерів тим, що він повинен діяти як шлюз між мережею Honeynet і примірником управління EC2. Для цього потрібно, щоб контейнер Honeywall мав два віртуальних мережевих інтерфейси: один в мостовій мережі для зв'язку з примірником хоста за замовчуванням, а інший – в мережі dmz.

Налаштувати контейнер Honeywall як шлюз для мережі dmz було не так просто, як здавалося на перший погляд. Виникли наступні проблеми:

- IP-адреса шлюзу повинна бути зазначена при створенні мережі; Неможливо вказати мережевий компонент (контейнер Honeywall) як шлюз;
- У контейнера не може бути IP-адреси для мережі, яка ще не була створена. Тому контейнеру не може бути спочатку присвоєно IP-адресу, а потім цю IP-адресу вказано як шлюз при створенні мережі.

Це означає, що мережа вже повинна бути визначена за допомогою шлюзу, перш ніж контейнер зможе стати її частиною. Зрештою, проблему було вирішено шляхом ручного управління правилами *iptables* і конфігурацією таблиці маршрутизації після запуску контейнера.

Щоб ця конфігурація була постійною при запуску контейнера в майбутньому, був створений новий образ Dockerfile, який містив ті ж параметри конфігурації, що і інші контейнери Cowrie, але використовував інший *entrypoint* сценарій, який налаштовував правила iptables і таблицю маршрутизації.

3.3.3 Проектування нової приманки з високим рівнем взаємодії

Під час налаштування контейнера Honeywall було виявлено, що Cowrie немає можливостей вихідної мережі, тому він не може встановлювати вихідні з'єднання SSH або telnet. Цей факт не був відомим на початку дослідження,

оскільки до реалізації системи було зрозуміло, що приманка Cowrie дійсно володіє цією функціональністю, оскільки відомо, що відправлення запиту *wget* всередині приманки завантажує запитаний ресурс і обчислює його контрольну суму SHA. Така функціональність потребує, щоб приманка встановила вихідне мережеве з'єднання, і передбачалося, що той же підхід буде застосований до реалізації служб SSH і telnet.

Коли це стало зрозуміло, початковий підхід до реалізації мережі Honeynet потребував негайної переоцінки. Можливість створення вихідних мережевих підключень з контейнера Honeywall необхідна, щоб зловмисник міг отримати доступ до Honeynet. Однак контейнер Honeywall також повинен являти собою максимально обмежене середовище, щоб мінімізувати ризик поширення атаки за межі системи. Ці міркування мотивували повну переоцінку конструкції системи.

З огляду на час, протягом якого проводилося дослідження, було б неможливо розширити вихідний код Cowrie, щоб дати приманці можливість встановлювати вихідні мережеві підключення по SSH і telnet. Замість цього було прийняте рішення запровадити нову Docker приманку з високим рівнем взаємодії для полегшення поширення атак за межами Honeywall: компонент, який далі буде називатись контейнером маршрутизатора (router).

Як приманка з високим рівнем взаємодії, контейнер маршрутизатора зіткнувся з безліччю нових проблем:

- Оскільки зловмисники більше не будуть взаємодіяти з емульованим середовищем всередині контейнера Docker, необхідно буде надати реальні бібліотеки та набори інструментів, щоб спонукати їх взаємодіяти з системою.
- Додаток Cowrie надає просту опцію конфігурації для вказівки комбінацій пароля та імені користувача, які повинні бути дозволені або заборонені для аутентифікації. Це не так просто відтворити в реальній системі, де обліковий запис користувача може бути налаштований тільки для

прийняття однієї правильної комбінації або при відсутності пароля для аутентифікації входу в систему.

- Зловмисники завжди будуть прагнути отримати root-доступ до системи, щоб мати над нею необмежений контроль. Приманка Cowrie ефективно емулювала цей доступ, але він не може бути сфальсифікований в реальній системі та потребує надання справжніх привілеїв *root* атакуючому. Це дуже ризиковано, оскільки немає жодних обмежень на дії, які зловмисники зможуть потім виконувати всередині системи.
- Для логування даних атаки необхідно буде визначити підхід до реєстрації мережевого трафіку і виконання команд, оскільки ці можливості Cowrie більше не будуть доступні для контейнера також.

Оскільки вимоги до контейнера маршрутизатора істотно відрізняються від вимог зі строгими обмеженнями до контейнера Cowrie, виникла необхідність в створені нового образу Docker.

Для того, щоб отримати доступ до широкого спектра потужних утиліт Linux, як основу для образу контейнера маршрутизатора було вирішено використовувати образ операційної системи Docker ubuntu:latest. Контейнеру також необхідно відкрити порти 22/TCP і 23/TCP, як і в контейнерах Cowrie, тому вони були вказані в Dockerfile за допомогою команди *EXPOSE*.

Найбільш загальні утиліти Linux, які були частиною образу Docker-Cowrie, також входили в цей новий образ: зокрема, пакети *apt-utils* і *build-essential,* які необхідні як базові утиліти майже у всіх Linux-системах на основі Debian.

Після визначення Dockerfile, для використання нового образу контейнера маршрутизатора замість старого образу контейнера honeywall було оновлено сценарій *do_honeypot.sh*. Аргументи *docker create* для додавання контейнера в мережу dmz і публікації портів 22/TCP і 23/TCP на інтерфейсі хоста залишилися без змін, оскільки контейнеру маршрутизатора все одно буде потрібна ця конфігурація.

Оскільки емуляція SSH і telnet була відсутня, наступним кроком стало встановлення і налаштування їх утиліт в контейнері маршрутизатора. Встановлення утиліти *openssh-server* було вказане в Dockerfile. Потім використовувався *entrypoint* сценарій для оновлення конфігурації SSH-демона *sshd*, щоб полегшити вхід в контейнер. Деякі специфічні конфігурації для забезпечення легкого доступу зловмисників до контейнера, які заслуговують на увагу, описані нижче:

- поле *PermitRootLogin*, яке було оновлено, щоб дозволити SSH-входи в систему як користувач root;
- Поле *PasswordAuthentication* було оновлено, щоб дозволити аутентифікацію за паролем, а не тільки з використанням ключа шифрування;
- Поле *PermitEmptyPasswords* було оновлено, щоб дозволити аутентифікацію з порожнім паролем, якщо це було допустимо для певного користувача.

Нарешті, *entrypoint* сценарій був оновлений для перезапуску, щоб конфігурація набрала чинності.

Інсталяція утиліт пакетів telnet – *openbsd-inetd* і *telnetd* також була додана в Dockerfile. В *entrypoint* сценарій був доданий ряд кроків налаштування, призначених для застосовування під час виконання контейнера: зокрема, *ttys*, який дозволяє проводити кілька одночасних сеансів telnet.

Файл конфігурації telnet, який включав ряд важливих параметрів конфігурації і ведення журналу, також був доданий в контейнер. Як і в разі конфігурації SSH, демон telnet потребував перезапуску, щоб зміни набрали чинності. У випадку пристроїв IoT, можливість ідентифікації типу та моделі пристрою дуже важлива для зловмисника при визначенні цілі вразливою, тому контейнер маршрутизатора повинен бути здатний надавати таку інформацію. Існує два основних повідомлення, які відправляються тим, хто ініціює з'єднання аутентифікації користувача з системою на основі Linux: файли *issue.net* і *mot.d*:

- *issue.net/issue* використовується для встановлення банеру, який відправляється при узгодженні протоколу. Наприклад, для клієнта, що відправляє запит на підключення до системи через telnet, це буде інформація, яка використовується для привітання перед етапом входу в систему. Для SSH цей файл називається *issue*, тоді як для telnet – *issue.net*.
- *motd/legal* використовується для зображення повідомлення клієнту після того, як він успішно увійшов до системи. Часто адміністратори використовують це для подання повідомлення про дозволене використання системи після того, як користувач успішно увійшов в систему. Для SSH цей файл називається *legal*, тоді як для telnet файл називається *motd*.

Команди для копіювання налаштованих файлів Issue.net, Issue, Legal та MOTD були додані У Dockerfile для контейнера маршрутизатора. Щоб ці банери представлялись в узгодженнях про з'єднання, були внесені деякі додаткові оновлення в *entrypoint* сценарій. На рисунку 3.8 зображено спробу входу через Telnet до примірника сервера-приманки. Як можна помітити, банер, який рекламує пристрій Huawei, представляється як частина запиту на вхід в систему.

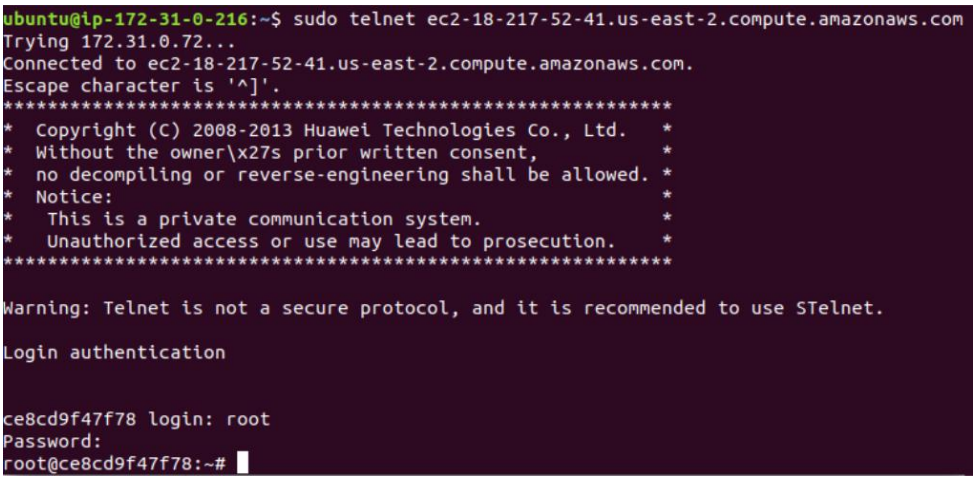

Рисунок 3.8 – Банер контейнера маршрутизатора

Для того, щоб зловмисник міг взаємодіяти з системою, він повинен бути переконаний в тому, що отримав доступ на рівні root до контейнера маршрутизатора. Однак, це без використання емульованого середовища, як

Cowrie, це неможливо. Тому було прийнято рішення дозволити *root*-доступ за замовчуванням для всіх облікових записів в цьому контейнері.

Пов'язана з цим проблема полягає в тому, що довільні комбінації паролів імені користувача не можуть бути прийняті таким же чином в реальному середовищі Linux, як і в імітованому середовищі Cowrie:

- Щоб зловмисник міг увійти в контейнер маршрутизатора під конкретним користувачем, для нього повинен існувати обліковий запис.
- Крім цього, для кожного облікового запису можна вказати тільки один дійсний пароль.

Реальність цього полягала в тому, що число комбінацій імені користувача і пароля, які можуть бути прийняті для приманки, є сильно обмеженим.

Для розв’язання цих проблем було визначено кілька облікових записів з іменами користувачів, які зазвичай спостерігаються при атаках методом грубої сили, і надано кожному з них привілеїв *root* за замовчуванням. Оскільки точні комбінації імені користувача і пароля, які використовуються в системі, будуть визначені в ході експериментів, в рамках Dockerfile був створений єдиний обліковий запис для користувача admin.

Щоб надати привілеї *root* для облікових записів, вони повинні бути оголошені як користувачі *sudo*, тому установку пакета *sudo* було необхідно включити як частину Dockerfile. Файл *sudoers* був налаштований таким чином, щоб він надав привілеї кореневого рівня користувачеві admin для всіх утиліт в контейнері. Тепер ця конфігурація повинна дозволяти зловмиснику, який авторизувався як адміністратор, успішно виконувати необмежені дії всередині контейнера.

Як і у випадку з контейнерами-приманками Cowrie, важливо, щоб контейнер маршрутизатора міг відстежувати дії зловмисника в системі. Цей контейнер являє собою приманку з широкими можливостями взаємодії: повноцінний контейнер з реальним мережевим стеком, наборами інструментів і файловою системою.

Найбільш часто використовуваною утилітою реєстрації подій в мережіє syslog. Syslog – це стандартна утиліта журналювання Linux IETF, яка реєструє мережеві і процесні події, що відбуваються в системі. Вона активно використовується системними адміністраторами для управління системою і аудиту безпеки [88]. Syslog було обрано як рішення для ведення журналу для збору інформації про з'єднання зловмисника з контейнером маршрутизатора, оскільки воно здатне збирати інформацію про реєстрацію як для SSH, так і для telnet.

Для того, щоб використовувати syslog, в Dockerfile був доданий пакет rsyslog. Потім в образ контейнера був доданий призначений для користувача файл конфігурації з ім'ям rsyslog.conf, і в *entrypoint* сценарії для контейнера була запущена утиліта rsyslog.

Щоб зберігати журнали, згенеровані rsyslog всередині контейнера, в контейнер маршрутизатора в каталозі */var/log* був змонтований. Події SSH і telnet реєструватимуться в декількох різних файлах журналу в цьому каталозі, включаючи auth.log, messages, secure і syslog.

Додатковим рішенням для моніторингу дій зловмисників всередині контейнера маршрутизатора повинно було стати використання Logkeys – кейлоггера Linux з відкритим вихідним кодом. [89] Однак, переважно через несумісність з Docker це виявилось неможливим. Тому була зроблена спроба ручного кейлогінгу, натхненна рекомендацією на онлайн-форумі askubuntu.com [91]. Цей підхід включав використання чисто командних оболонок, згенерованих як частина реєстрації мережевих подій системного журналу. Зрештою, він був визнаний ненадійним і крихким.

Останнім етапом стало визначення початкових утиліт, які необхідно було додати в Dockerfile для контейнера маршрутизатора: net-tools, nmap, traceroute, inetutils-ping, iptables і tcpdump. Вони всі є мережевими пакети, які дозволять зловмиснику виконувати такі дії, як перегляд і управління таблицею маршрутизації контейнера, сканування сусідніх хостів і захоплення пакетів в мережі тощо.

3.3.4 Перевірка конфігурації мережі Honeynet

Тепер, коли всі необхідні компоненти мережі Honeynet були реалізовані, залишилося тільки переконатися, що конфігурація мережі працює правильно. По замовчуванню зловмисник повинен мати вільний доступ до контейнера маршрутизатора з Інтернету і взаємодіяти з будь-якими доступними контейнерами Cowrie в мережі dmz. Як видно з рисунка 3.9, наявні дві мостові мережі: bridge і DMZ. Примірник хоста відправляє свій власний мережевий трафік через порти 22/TCP та 23/TCP на віртуальний інтерфейс *docker0* на мості bridge. Цей інтерфейс з'єднує хост з віртуальним інтерфейсом маршрутизатора контейнера маршрутизатора *eth0*. Контейнер маршрутизатора також має другий віртуальний інтерфейс *eth1*, який з'єднує його з мережею dmz. Саме через нього стає можливим доступ до приманок Cowrie.

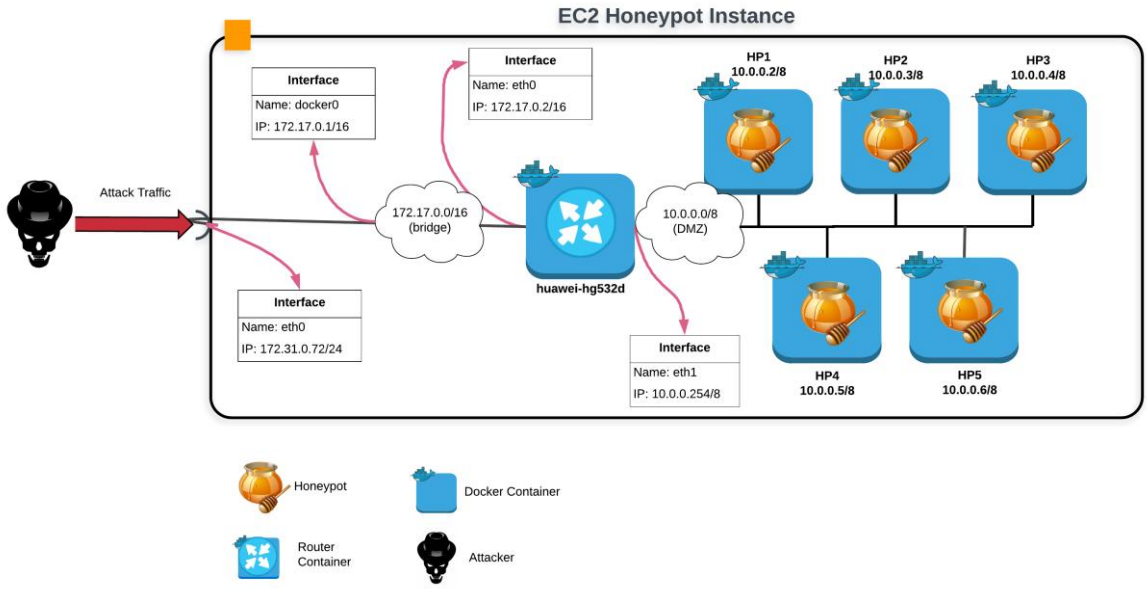

Рисунок 3.9 – Конфігурація мережі Docker на примірнику-приманці

Щоб контейнер маршрутизатора був доступний з Інтернету, необхідно налаштувати його конфігурацію. Для цього спершу необхідно перевірити налаштування мостової мережі та таблицю маршрутизації на самому примірнику EC2. Вимоги до конфігурації проілюстровані на рисунку 3.10.

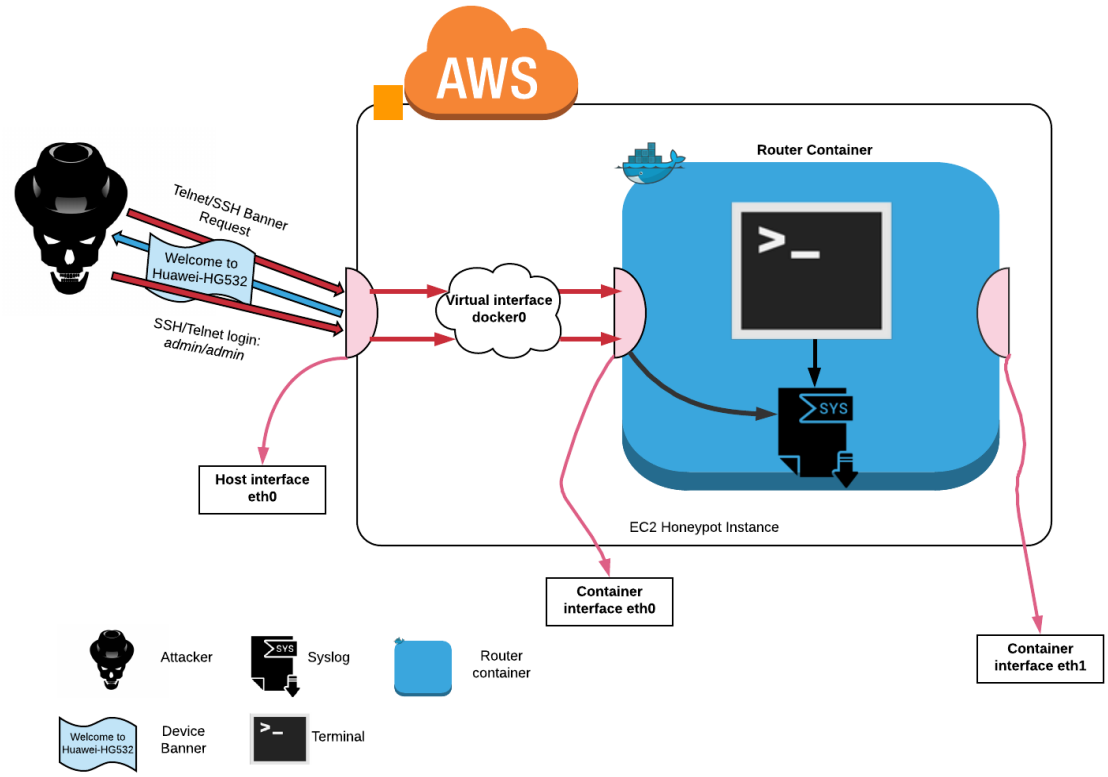

Рисунок 3.10 – Маршрутизація трафіку атаки від межевого інтерфейсу хоста до контейнера маршрутизатора

Як показано на рисунку, зловмисник робить спробу аутентифікації методом грубої сили через SSH або telnet, яка перенаправляється з інтерфейсу реального хоста *eth0* на віртуальний інтерфейс *docker0* в мережі моста. З цього інтерфейсу трафік прямує в інтерфейс контейнера маршрутизатора *eth0*. Щоб відповісти зловмисникові, мережевий трафік відправляєтесь через ці інтерфейси у зворотному порядку.

Внутрішня таблиця маршрутизації примірника приманки EC2, в якій показано мережі, пов'язані з його фізичним мережевим інтерфейсом *eth0* і віртуальним мережевим інтерфейсом Docker *docker0*, можна бачити в таблиці 3.1.

Таблиця 3.1 – Внутрішня таблиця маршрутизація примірника приманки

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Interface
0.0.0.0	172.31.0.1	0.0.0.0	UG	0	0	0	*eth()*
172.17.0.0	0.0.0.0	255.255.0.0	U	0	0	0	*docker()*
172.31.0.0	0.0.0.0	255.255.240.0	U	0	0	0	*eth()*

Як можна спостерігати з таблиці, є два інтерфейси: *eth0* і *docker0*. *eth0* підключає хост до зовнішньої мережі 172.31.0.0/24, яка керується AWS. Інтерфейс *docker0* відповідає мостовій мережі з адресою 172.0.0.0/16, до якої за замовчуванням підключені контейнери.

За своєю конструкцією хост не може безпосередньо взаємодіяти з мережею моста dmz: як обговорювалося в розділі 2.2.4, запобігаючи підключенню хоста і контейнерів до непотрібних мереж, ризик компрометації цих систем при атаці зводиться до мінімуму.

Виконавши команду *Docker Network Inspect Bridge*, можна перевірити властивості мостової мережі Docker. Результат виконання цієї команди показаний на рисунку 3.11:

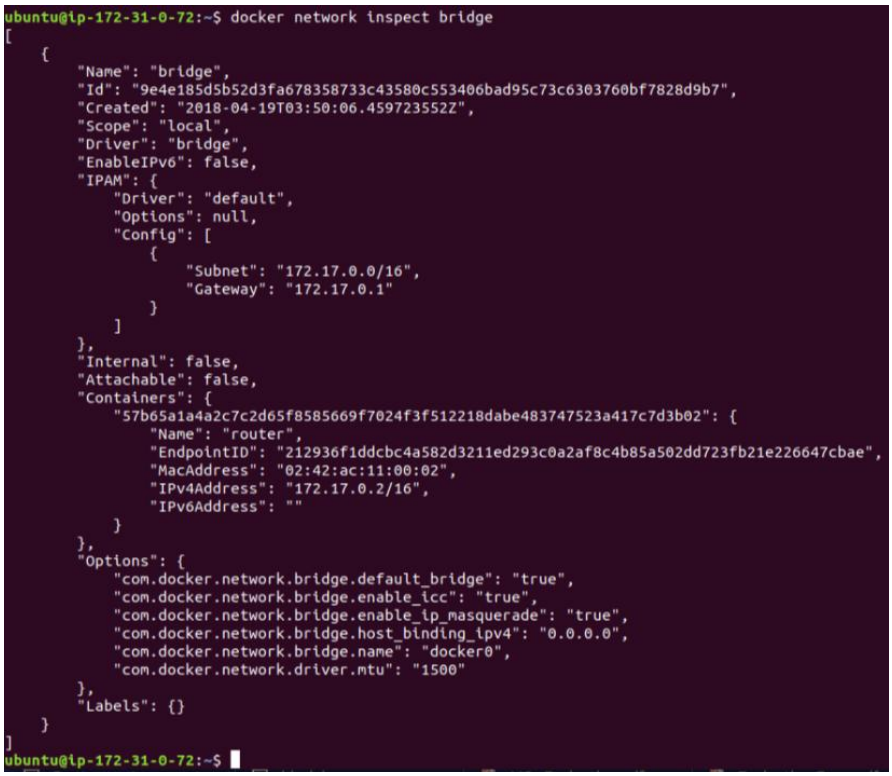

Рисунок 3.11 – Результат виконання консольної команди *Docker Network Inspect Bridge*

Якщо згодом виконується команда *docker container inspect router*, отриманий результат демонструє такі очікувані властивості:

- Контейнер маршрутизатора підключений до мостової мережі з IP-адресою 172.17.0.2/16;

- Контейнер маршрутизатора також підключений до мережі DMZ з IP-адресою 10.0.0.254/8;
- Порти 22/TCP і 23/TCP контейнера маршрутизатора публікуються на порти 22/TCP і 23/TCP інтерфейсу хоста.

Правильну маршрутизацію трафіку атаки через мережу bridge можна перевірити ззовні примірника приманки, якщо спробувати підключившись до нього через SSH або telnet і перевірити, чи відображається запрошення входу в контейнер маршрутизатора. На рисунку 3.12 показано вивід консолі при підключенні SSH до примірника сервера-приманки, який демонструє правильну маршрутизацію цього трафіку в контейнер маршрутизатора.

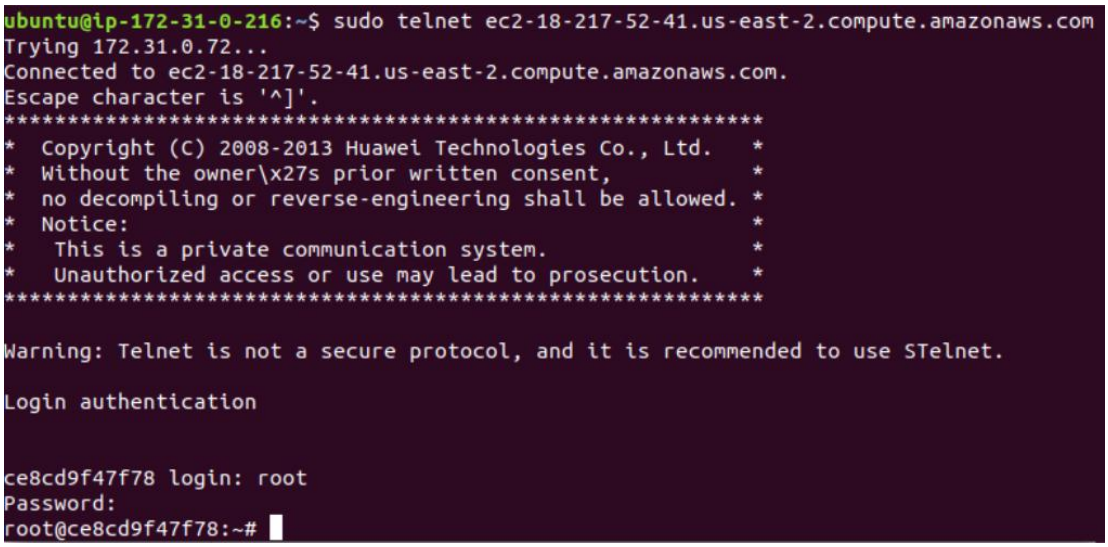

Рисунок 3.12 – Спроба входу по SSH в контейнер маршрутизатора

Щоб переконатися, що трафік атаки може досягати контейнерів Cowrie, необхідно перевірити конфігурацію контейнера маршрутизатора та мережі dmz. Зловмисник, опинившись всередині контейнера маршрутизатора, повинен мати можливість під’єднатися через інтерфейс на контейнері маршрутизатора до будь-якого контейнера Cowrie в мережі dmz. Припускаючи, що зловмисник вже успішно скомпрометував контейнер маршрутизатора за допомогою процесу, показаного на рисунку 3.10, він повинен мати можливість підключитися до контейнера Cowrie через SSH або telnet таким же чином через інтерфейс eth1

контейнера маршрутизатора в мережі dmz. Вимоги до маршрутизації на контейнер приманки показані на рисунку 3.13.

Внутрішню таблицю маршрутизації контейнера-маршрутизатора можна побачити в таблиці 3.2

Таблиця 3.2 – Внутрішня таблиця маршрутизації контейнера router

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Interface
0.0.0.0	172.17.0.1	0.0.0.0	UG	0	0	0	*eth()*
10.0.0.0	10.0.0.254	255.0.0.0	U	0	0	0	*eth1()*
172.17.0.0	0.0.0.0	255.255.0.0	U	0	0	0	*eth()*

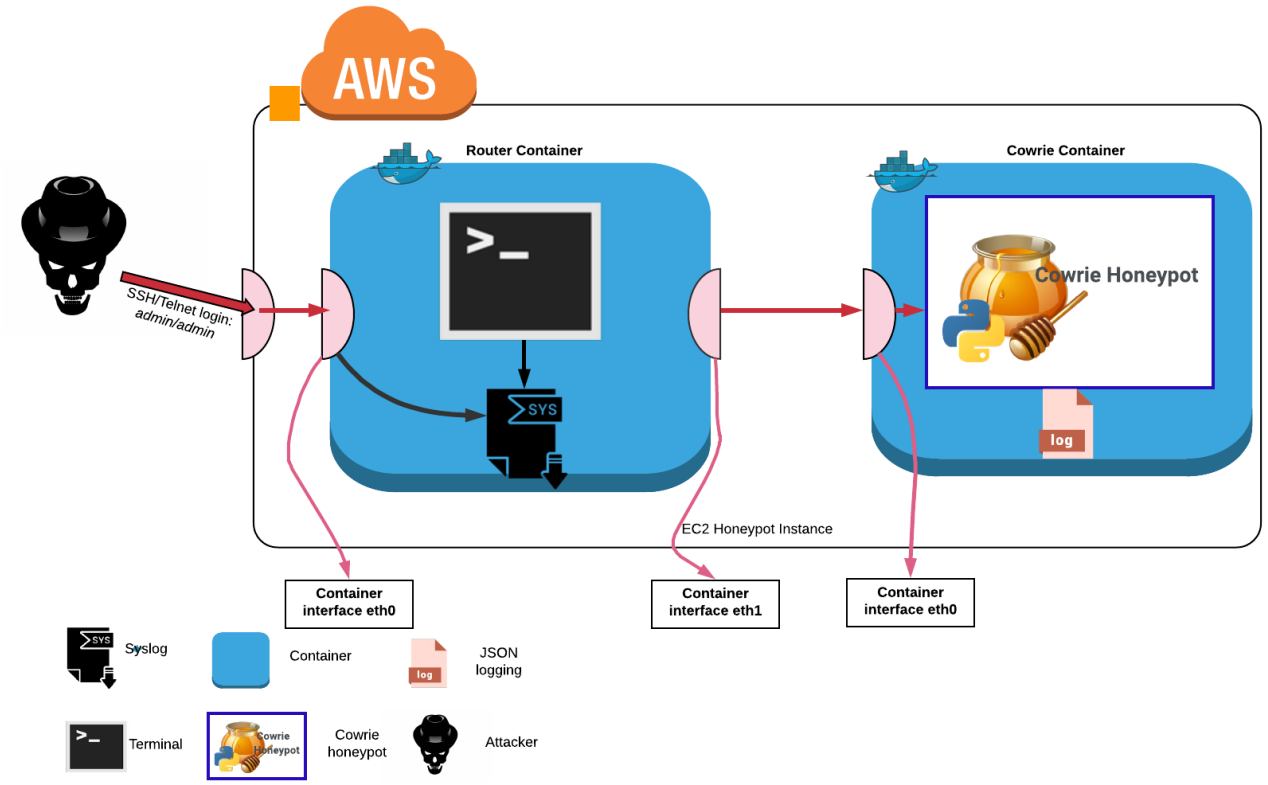

Рисунок 3.13 – Маршрутизація мережевого трафіку атаки на контейнер Cowrie

Зсередини контейнера маршрутизатора зловмисник повинен мати доступ до зовнішніх доменів Інтернету, щоб виконувати завантаження для просувати свою атаку на систему. Це досягається шляхом налаштування правил перетворення мережевих адрес (NAT) в *iptables*. Правила, які дозволяють

зловмиснику виконувати завантаження з зовнішніх доменів, показані у лістингу 3.3.

Лістинг 3.3 – Правила iptables, які були налаштовані для включення NAT всередині контейнера маршрутизатора

```
iptables - -table - -append POSTROUTING
     - -out-interface th0 -j MASQUERADE

iptables - -append FORWARD - -interface eth1 -j ACCEPT
```

Таблицю маршрутизації, яка була згенерована всередині контейнера Cowrie, можна побачити в таблиці 3.3. В цьому випадку контейнер підключений лише до мережі dmz, до якої він звертається через свій віртуальний інтерфейс eth0.

Таблиця 3.3 – Таблиця маршрутизації, створена командою route -n, яка виконана всередині контейнера Cowrie

Destination	**Gateway**	**Genmask**	**Flags**	**Metric**	**Ref**	**Use**	**Interface**
0.0.0.0	1.0.0.254	0.0.0.0	UG	0	0	0	*eth()*
10.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	*eth()*

Виконавши команду *Docker Network Inspect dmz*, можна перевірити властивості мережі Docker dmz. Результат виконання цієї команди показаний на рисунку 3.14.

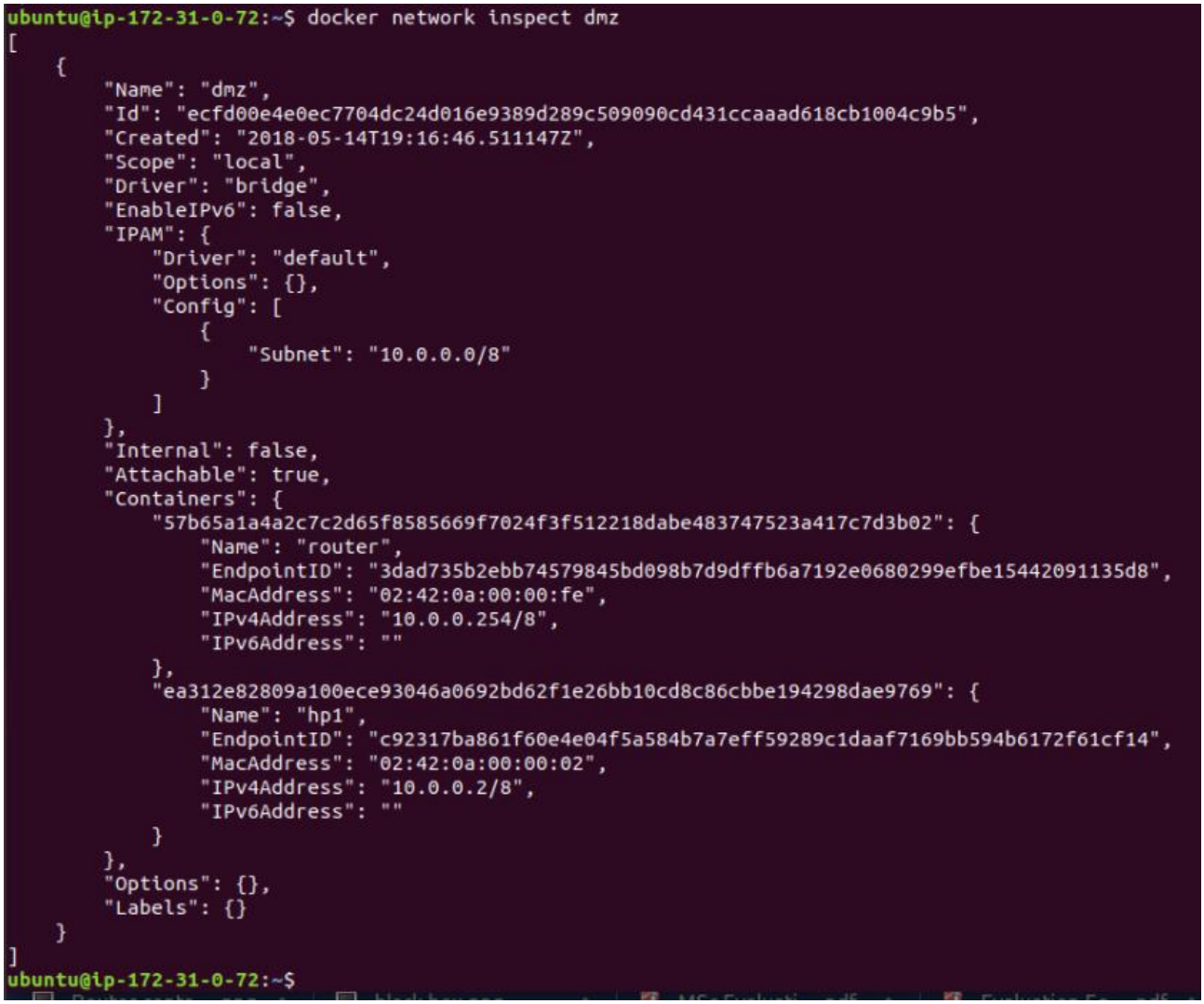

Рисунок 3.14 – Результат консольної команди *docker network inspect dmz*

Як можна бачити на рисунку, в цій мережі є 2 контейнери: контейнер маршрутизатора та контейнер Cowrie (hp1) відповідно. Якщо виконати команду *docker container inspect hp1*, можна спостерігати наступне:

- Контейнер hp1 має IP-адресу 10.0.0.2/8 в мережі dmz;
- Порти 22/TCP і 23/TCP контейнера hp1 доступні в мережі, але не мають прив'язки до хосту.

Правильна маршрутизація трафіку атаки від контейнера маршрутизатора через dmz до контейнера hp1 Cowrie може бути перевірена зсередини контейнера маршрутизатора. Опинившись в контейнері маршрутизатора, підключення по SSH або telnet до приманки hp1 і відображення запрошення на вхід в Cowrie перевіряє правильність конфігурації мережі. На рисунку 3.15

показаний вивід консолі при спробі підключення SSH від контейнера маршрутизатора до контейнера hp1, який демонструє правильну маршрутизацію трафіку в мережі dmz.

Рисунок 3.15 - Відмова в SSH-вході від маршрутизатора до примірника приманки Cowrie

З рисунка можна помітити, що «зловмисник» намагається увійти в контейнер hp1 як користувач admin. Перший пароль, наданий для входу в систему, відхиляється, а другий запит пароля запитується вже приманкою hp1 Cowrie. В цьому випадку зловмисник вирішує скасувати спробу входу.

3.4 Візуалізація даних атаки

Як було вирішено в розділі 2.1.2, дані, що генеруються приманками в Honeynet Docker, будуть візуалізуватись з використанням стека обробки журналу ELK на окремому примірнику сервера EC2. Розгортання примірника сервера управління EC2 вже розглядалось, тому в цьому розділі описується лише встановлення і налаштування інструментів для забезпечити обробку та візуалізацію даних.

3.4.1 Візуалізація на примірнику сервера управління

Примірник сервера управління EC2 вже було розгорнуто з достатніми апаратними ресурсами для відповідності вимогам стеку ELK. Оскільки стек

ELK широко використовується для обробки та візуалізації журналів, було доступно безліч ресурсів для ознайомлення, які показують, як налаштувати ці інструменти для спільної роботи.

Щоб полегшити використання стеку ELK для аналізу журналів з віддаленого комп'ютера також використовувалися деякі додаткові інструменти: зокрема, Nginx і Filebeat. Їх конфігурація і взаємодія описуються далі.

Elasticsearch - це пошуковий і аналітичний движок з відкритим вихідним кодом, заснований на Apache Lucene API. Той факт, що він підтримує аналіз файлів у форматі JSON, зробив його ідеальним вибором для аналізу та індексації журналів Cowrie.

Щоб налаштувати середовище для запуску Elasticsearch, як обов'язковий компонент спочатку була встановлена Java 8. Далі, для безпечного налаштування Elasticsearch необхідно було обмежити зовнішній доступ до додатка. Це важливо, оскільки Elasticsearch надає HTTP API, які можна запитувати для виконання маніпуляцій з даними, що зберігаються в індексі. Якщо до цього доступ не був обмежений, дані в індексі можуть бути змінені або видалені без дозволу.

Змінюючи файл конфігурації /etc/elasticsearch/elasticsearch.yml для Elasticsearch, зовнішній доступ був повністю обмежений, а Адреса доступу localhost була змінена на network.host. Це означало, що доступ до Elasticsearch можливий тільки з самого примірника управління на локальному хості через 9200/TCP порт за замовчуванням.

Механізм візуалізації Kibana є другим компонентом стека ELK, який був встановлений і налаштований. Kibana - це платформа для аналізу та візуалізації з відкритим вихідним кодом, розроблена спеціально для роботи з пошуковою платформою Elasticsearch. Використання Kibana в цьому проекті було направлено на те, щоб зробити аналіз журналів приманки простим і негайним.

Kibana надає веб-інтерфейс, через який можна отримати доступ до візуалізацій. Налаштовуючи правила для візуалізації вхідних даних журналу, Kibana дозволяє миттєво розуміти важливу інформацію про стан системи.

Як і в Elasticsearch, було важливо налаштувати Kibana так, щоб вона була обмежена для доступу з зовнішніх систем. Для досягнення цієї мети був використаний інструмент зворотного проксінгу Nginx.

Nginx - це інструмент з відкритим вихідним кодом, орієнтований на надання таких послуг, як веб-обслуговування, зворотний проксінг, кешування і балансування навантаження [92]. У цій системі він використовувався для надання доступу до веб-інтерфейсу Kibana з використанням зворотного проксі. Крок аутентифікації був налаштований з використанням єдиного дійсного імені користувача та пароля, щоб забезпечити автентифікований вхід в систему з веб-браузера. Також була потрібне деяке додаткове налаштування для направлення вхідного HTTP-трафіку в додаток Kibana.

Logstash - це інструмент конвеєра журналів з відкритим вихідним кодом, який приймає, обробляє, перетворює і виводить дані з файлів журналів, наданих йому. Він був останнім компонентом, який налаштовувався на примірнику сервера управління.

Як обговорювалося в розділі 2.1.2, сертифікат SSL був створений з використанням приватної IP-адреси сервера в полі SAN. Пізніше це буде використовуватися Logstash для перевірки походження даних журналу, що відправляються Filebeat.

Щоб налаштувати компоненти обробки Logstash, необхідно визначити ряд конфігурацій для роботи з кожним з форматів вхідного журналу: конфігурації введення, фільтрації і виведення. Всі вони були вказані як частина одного і того ж файлу, 03-cowrie.conf.

- Конфігурація введення

Введення – це поле, яке використовується для передачі даних в конвеєр Logstash. У цій системі це поле було визначене для отримання даних типу ударів, які відповідають даним, відправленим Filebeat.

Був вказаний номер порту, з якого ці дані повинні бути отримані Filebeat. Також було вказано шлях до згенерованого сертифіката SSL, щоб дані, отримані на цьому порту, могли бути перевірені Logstash.

- Конфігурація фільтр

Фільтри є проміжним етапом обробки в конвеєрі Logstash, в якому операції можуть застосовуватися до вхідних даних. Було знайдено відмінний підручник, в якому пояснюється ряд різних операцій фільтрації, які можна застосовувати до журналів Cowrie. Вони лягли в основу операцій фільтрації, налаштованих в цьому полі [93].

- Конфігурація виведення

Виведення є останнім полем в конвеєрі обробки Logstash, у якому результати обробки передаються в призначений процес. У цьому випадку як процес виведення використовувався Elasticsearch, який був налаштований на прослуховування вхідних даних на локальному хості, порт 9200. Деякі додаткові параметри були вказані в відношенні форматування даних, які будуть зберігатися в індексі Elasticsearch.

3.4.2 Візуалізація на примірнику приманки

Щоб надати примірнику управління EC2 дані з приманки для візуалізації, журнали, згенеровані контейнерами honeypot на екземплярі honeypot EC2, їх необхідно об'єднати й потім безпечно передати в екземпляр управління. Підрозділи, описані нижче, пояснюють, яким чином це було досягнуто.

Для копіювання вмісту журналів використовувався інструмент CRON.

Лістинг 5.4. У цьому фрагменті показані 2 завдання, заплановані з використанням CRON, які копіюють вміст томів контейнера-приманки в каталог / var / logs. Ці завдання виконуються кожні 60 секунд (максимально можлива частота для завдання CRON).

```
# m h dom mon dow    command
* * * * * /bin/cp -a   /home/ubuntu/cowrievolumes/*/
     /var/log/cowrie/ >> /var/log/cowrie/cronlog.log
* * * * * /bin/cp -a /home/ubuntu/router/*/
     /var/log/cowrie/router/ >> /var/log/cowrie/cronlog.log
```

В якості ще одного важливого елемента монітора кібер-інцидентів, визначеного в розділі 2.1.2, є налаштування системи сповіщень про загрозу.

3.5 Налаштування системи сповіщень про загрозу

Як пояснено в розділі 2.1.2**,** PSAD був визначений як інструмент вибору, що дозволяє попереджати системних адміністраторів про потенційні атаки. Базова конфігурація генерувала оповіщення по електронною поштою при виявленні примірника Honeypot, відправляючи повідомлення на адресу hdstkr@outlook., яку можна вказати у файлі конфігурації /etc/psad/psad.conf. Перевірку стану журналів мережевих подій PSAD можна налаштувати так часто, як це необхідно. Приклад оповіщення зображено на рисунку 3.16. При скануванні порту було згенеровано сповіщення електронною поштою, показане в правій панелі поштової скриньки. PSAD виявив, що джерелом сканування є домен io-in-f103.1e100.net, який відправив кілька ICMP-пакетів на примірник сервера EC2, що і викликало це попередження.

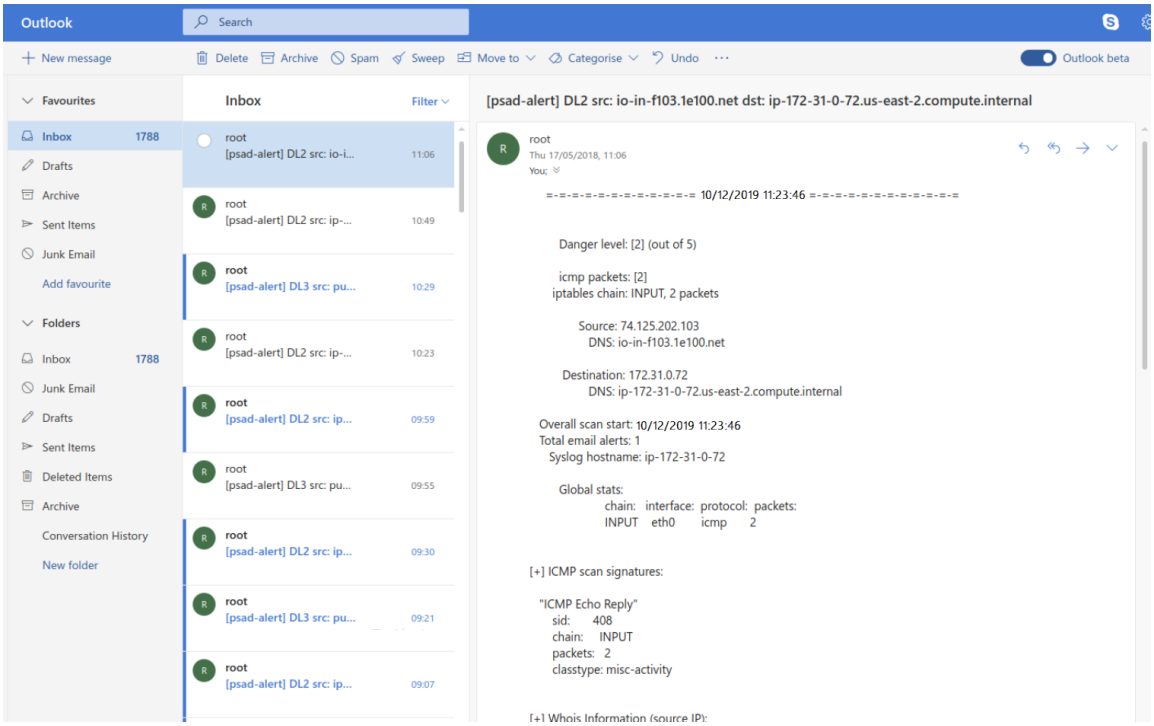

Рисунок 3.16 - Оповіщення електронною поштою від PSAD

За допомогою PSAD також можна повністю внести в чорний або білий список певні IP-адреси, відредагувавши файл конфігурації /etc/psad/auto_dl. Ця функціональність була використана для зменшення кількості помилкових спрацьовувань, які генеруються PSAD.

3.6 Висновок до розділу

На той час, коли всі компоненти системи були сконфігуровані та реалізовані, як описано в цьому розділі, були досягнуті такі цілі:

- Сеанси атаки можна змоделювати, підключившись до примірника приманки EC2 по SSH або telnet і представивши його в запрошенні на вхід в контейнер маршрутизатора. Після аутентифікації в контейнері маршрутизатора стало можливим аналогічним чином підключатися до будь-якого контейнера в мережі Honeynet і взаємодіяти з приманками Cowrie.
- Вся згенерована діяльність із взаємодій з контейнерами Cowrie була успішно відправлена, оброблена і візуалізована за допомогою панелі візуалізації, яка доступна з веб-браузера.
- При необхідності, все середовище Docker може бути повністю вилучене і повторно розгорнуте з ідентичною конфігурацією.

Таким чином, мета розробки монітора кібер-інцидентів, який керується приманкою, була значною мірою досягнута. При цьому залишається багато можливостей для налаштування як приманок, так і візуалізацій даних про атаки. У наступному розділі ця система буде використовуватися для проведення серії експериментів, спрямованих на визначення ефективної структури адаптивних приманок.

4 ОЦІНКА РЕАЛІЗОВАНОЇ СИСТЕМИ HONEYNET

В цьому розділі основна увага приділяється оцінці реалізованої системи моніторингу кіберінцидентів, а також опису планування та проведення експериментів як частини пропозиції щодо проектування адаптивної приманки.

4.1 Планування експериментів

Проведення експериментів сфокусовано на досягненні другої мети, викладеної в розділі 1.5.2: визначити покращену структуру для ефективних адаптивних приманок. Було вирішено, що міра ефективності в цих експериментах буде залежати від того, скільки атак отримала кожна з приманок. Виходячи з цього, згодом можна буде надати результати та зробити висновки щодо розробки ефективних адаптивних приманок.

На планування експериментів великий вплив справив час, який залишився до завершення дослідження: період всього чотири тижні. Протягом них була розроблена серія ітерацій експерименту. У таблиці 4.1 описані змінні, що беруть участь в кожній ітерації, і передбачуване виконання цих ітерацій щодо часу.

Таблиця 4.1 – Набір запланованих ітерацій експерименту

	Виробник А				**Виробник Б**				**Виробник В**				**Перевірчий**			
	Тиждень 1				*Тиждень 2*				*Тиждень 3*				*Тиждень 4*			
	LC	**SA**	**DT**		**LC**	**SA**	**DT**		**LC**	**SA**	**DT**		**LC**	**SA**	**DT**	
HP1	A	X	P		A	X	P		A	X	P		A	X	P	
HP2	B	Y	Q		B	Y	Q		B	Y	Q		B	Y	Q	
…	C	Z	R		C	Z	R		C	Z	R		C	Z	R	
HP_n	D	W	S		D	W	S		D	W	S		D	W	S	
Перевірчий	*	*	*		*	*	*		*	*	*		*	*	*	

Розглядаються три характеристики приманок Cowrie HP1, HP2 ... HPN: прийняті облікові дані для входу (LC), оголошені служби для роботи в контейнері (SA) і тип оголошеного пристрою (DT). Для кожного тижня змінюється лише одна характеристика приманки-маршрутизатора: постачальник пристрою, який відповідає стовпцям Виробник А, Виробник Б і Виробник В. Кожна варіація характеристики приманки Cowrie (LC, SA, DT) повинна існувати протягом 2 днів разом з усіма 12 ітераціями експерименту, що відбуваються протягом 4 тижнів. Для організації експериментів виділяється один день на 7-денний тиждень, який представлено порожнім стовпцем для кожного тижня. У кожній ітерації є одна приманка Control, яка відіграє роль показника базової продуктивності, а на 4 тижні з цією ж метою використовується і приманка маршрутизатора Control.

Пояснення того, як цей план перетвориться в конфігурацію дослідницького середовища, представлено в наступних підрозділах.

4.1.1 Налаштування контейнера маршрутизатора для експериментів

Очевидно, що контейнер маршрутизатора як шлюзу між Інтернетом і мережею Honeynet повинен допомогти зловмиснику отримати доступ до розгорнутої системи. Полегшення доступу до цього контейнеру, ймовірно, збільшить ймовірність успішного проникнення в мережу приманок, дозволяючи збирати цінні дані. Таким чином, було важливо налаштувати контейнер маршрутизатора так, щоб він був максимально привабливим і відкритим для цільових атакуючих.

Система вже була розроблена і впроваджена для задоволення саме цих вимог, тому контейнер маршрутизатора міг легко забезпечити:

- Віддалений доступ через SSH або telnet із загальнодоступного Інтернету;
- Аутентифікацію з використанням ряду допустимих комбінацій імені користувача і пароля;
- Кореневі привілеї всередині системи;

- Потужну ОС і інструментарій;
- Мережу сусідніх хостів, на які може поширюватися атака;
- Привабливий банер, який вказує на тип пристрою.

Щоб отримати деяке уявлення про конструкцію приманки з контейнера маршрутизатора, а не тільки з приманок Cowrie, в рамках експериментів було вирішено кожного тижня змінювати постачальника пристрою, який рекламується. Всі інші характеристики контейнера (облікові дані, сервіси, набори інструментів тощо) залишаться постійними і будуть доступні на цій приманці на всіх ітераціях експерименту.

З посиланням на план ітерацій експерименту в таблиці 4.1 були розроблені такі експерименти для вимірювання впливу постачальника пристрою на кількість отриманих атак:

1) Для тижня 1, тижня 2 і тижня 3, які показані в таблиці, за допомогою банерів SSH і telnet контейнер маршрутизатора буде налаштований для оголошення маршрутизаторів різних брендів. Кожен банер буде залишатися в контейнері маршрутизатора протягом 6 послідовних днів цього тижня. Всі інші змінні в середовищі приманки підтримуються постійними.

2) На 4-му тижні буде використовуватися перевірчий контейнер маршрутизатора: той, який не рекламує якийсь бренд або модель, використовуючи стандартний SSH Ubuntu 16.0.4 LTS і банер telnet з іменем хоста *router*. Використовуючи такий елемент контролю, можна порівнювати результати інших трьох тижнів з базовим рівнем, щоб визначити, чи має значення, що пристрій рекламується як той, який належить конкретному постачальнику.

Для того, щоб 3 банери пристроїв були налаштовані на цій приманці для тижня 1, тижня 2 і тижня 3, є бажаним, щоб бренди пристроїв, які використовувалися в конфігурації банера, були основані на реальних даних. Masscan – це потужний асинхронний сканер портів, який використовувався розробниками приманки IoTPot для захоплення банерів пристроїв з пристроїв в Інтернеті, які прослуховують порт telnet 23/TCP [58]. Ефективність цього

підходу в оцінці поведінки бот-мереж IoT мотивувала рішення використовувати Masscan для захоплення банерів пристроїв для використання в експериментах в цьому дослідженні. Як основа для експериментів будуть використовуватися банери з виробниками пристроїв, які найчастіше зустрічаються, оскільки можна припустити, що більш популярний постачальник буде піддаватися атаці з більшою ймовірністю, ніж менш популярний.

4.1.2 Налаштування контейнерів Cowrie

У мережі приманок Cowrie є значно більше можливостей для вимірювання впливу різних характеристик, оскільки є більша кількість доступних приманок: мотивована формулюванням цілей дослідження в розділі 1.5.2, мережа Honeynet забезпечує рішення для точного вимірювання відносної ефективності різних конфігурацій приманок в одному і тому ж середовищі.

Для кожного з 4 тижнів, протягом яких проводились експерименти, було вирішено, що 3 характеристики приманок будуть варіюватися протягом періоду 2 дні на тиждень:

- Облікові дані;
- Сервіси, що працюють на контейнерних портах;
- Тип пристрою, який рекламується.

Щоб проілюструвати, як це переходить до експериментальної ітерації, розглянемо як приклад стовпець LC на тижні 1 в таблиці 4.1.

- LC відповідає зміні облікових даних для входу в систему, які приймаються приманкою HP1, HP2... HP_n протягом 2-денного періоду.
- Для кожної з приманок дозволені облікові дані повинні відрізнятися. Інші змінні залишаються постійними.

В кожній ітерації експерименту буде залучено п'ять приманок Cowrie, одна з яких буде використана для перевірки. Наявність такої ітерації важлива, оскільки вона дозволяє зробити остаточні висновки щодо того, чи впливають

результати, отримані на налаштовані приманки, на кількість атак, які вони отримують.

4.2 Проведення експериментів

Планувалося, що ітерації експерименту будуть проводитися в порядку, зазначеному в таблиці 4.1, починаючи зі зміни облікових даних для входу в приманках Cowrie і додавання банера, який рекламує виробника пристрою.

4.2.1 Захоплення банерів пристроїв

Для сканування всього адресного простору IPv4 на наявність пристроїв, що прослуховують порти 22/TCP (SSH) або 23/TCP (telnet) протягом двох днів використовувалась утиліта Masscan. Банери telnet і SSH, а також IP-адреси цих пристроїв були зафіксовані в файли журналів.

Згодом банери були витягнуті з файлів журналу в новий файл, де їх було підраховано за допомогою shell-команд. Всього було просканово 70 137 банерів після сканування порту 23/TCP (telnet) і 164 291 після сканування порту 22/TCP (SSH).

Багато з отриманих банерів містили невизначену інформацію про модель і виробника пристроїв, причому багато з них не були ідентифіковані взагалі. Щоб зробити отримані результати відносно придатними для фальсифікації банерів на приманках, результати Masscan використовувався в поєднанні з результатами пошуку вразливих моделей пристроїв в Google.

За результатами сканування було визначено, що пристрої Huawei зустрічаються найчастіше: 43 рази в банерах telnet і 5 разів в банерах SSH. Це дозволило ідентифікувати Huawei як найпопулярнішого виробника маршрутизаторів, якого можна було розпізнати по захоплених банерах. Проте, оскільки по цих банерах не вдалось виявити ніяких конкретних моделей, було проведено додаткове дослідження вразливих моделей маршрутизаторів Huawei,

в результаті якого їх модель HG-532d була визначена як вразлива для використання [94];[95]. Також було виявлено, що в керівництві до пристрою містяться облікові дані користувача за замовчуванням. Таким чином, для першої ітерації експерименту контейнер маршрутизатора був оголошений як маршрутизатор Huawei HG-532d.

4.2.2 Експеримент 1, Спроба 1

Перша розпочата ітерація експерименту відповідала експерименту LC 1-го тижня, показаному в таблиці 4.1. Вона охоплює зміну облікових даних для входу в систему і налаштування банера пристрою конкретного виробника в контейнері маршрутизатора.

Щоб сприяти якомога більшій кількості успішних входів в систему, контейнер маршрутизатора був налаштований так, щоб дозволити вхід без пароля для 14 різних облікових записів користувачів. Ці облікові записи були додані на основі результатів, отриманих під час тимчасового розгортання базової приманки Cowrie на примірнику AWS EC2, який був налаштований для збору облікових даних, які фактично використовуються бот-мережами IoT для атаки на автентифікацію методом грубої сили.

Як вже було зазначено раніше, як частина цієї ітерації експерименту були розгорнуті п'ять приманок Cowrie. Однією з них була перевіряюча, базова приманка Cowrie без будь-яких додаткових налаштувань.

Оскільки облікові дані для входу в систему були типовими для решти 4 приманок в цій ітерації експерименту, тип обладнання та послуги, доступні на них, необхідно було підтримувати постійними.

- Оголошений тип пристрою

Всім приманкам були присвоєні імена пристроїв, аналогічні імені хоста за замовчуванням, які призначені Cowrie. Таким чином, приманки Cowrie були названі srv01, srv02, srv03, srv04 і srv05, з яких srv04 була управляючою.

- Рекламовані послуги

Найпростішим рішенням для збереження сервісів, оголошених приманкою, було використання конфігурації контейнера Cowrie за замовчуванням (емульовані служби SSH і telnet були єдиними сервісами, які могли бути виявлені зловмисником при скануванні).

- Облікові дані

Для кожної з приманок Cowrie (окрім srv04) були налаштовані різні облікові дані:

1) srv01 отримав ім'я користувача root з будь-яким паролем;

2) srv02 отримав ім'я користувача admin з будь-яким паролем;

3) srv03 отримав будь-які облікові дані, надані після випадкового числа спроб, які можна налаштувати за допомогою опцій *AuthRandom* в файлі *cowrie.cfg*;

4) srv05 використав комбінації імені користувача та пароля, отримані в розділі 3.1.2;

Після того, як система була відкрита для Інтернету через порти 22/TCP (SSH) і 23/TCP (telnet), відразу ж було виконано безліч спроб підключення, які захопив контейнер маршрутизатора. Однак, незабаром, за відмітками часу подій було виявлено, що більшість з’єднань розірвано відразу після підключення.

Після деяких роздумів був зроблений висновок про те, що така поведінка може бути пов'язаною з виявленням системи як середовища приманки: оскільки зловмисникам надано root-доступ без пароля до контейнера маршрутизатора, було б розумно очікувати, що це викличе підозри щодо легітимності системи. Це не оптимально, оскільки мета надання доступу без пароля полягала в тому, щоб дозволити зловмисникам легко отримати доступ до системи. Налаштування аутентифікації по паролю, яка допускає тільки 1 пароль на користувача системи, обмежило б число зловмисників, які могли успішно отримати доступ система.

Ітерація експерименту була завершена через 2 дні. До цього часу жодна з приманок Cowrie в підмережі не реєструвала ніяких атак, оскільки жоден з

атакуючих не пройшов далі контейнера маршрутизатора. На цій підставі було вирішено, що ітерація експерименту повинна повторюватися при тих самих умовах, за винятком того, що до контейнера маршрутизатора більше не має надаватися доступ без пароля.

4.2.3 Експеримент 1, Спроба 2

На підставі того, що система не змогла зібрати будь-які суттєві дані атаки в першій спробі експерименту, система була повторно розгорнута з тією ж конфігурацією. Однак на цей раз автентифікація без пароля в контейнері маршрутизатора заборонена.

Як альтернатива автентифікації без пароля було вирішено, що пароль для кожного облікового запису буде налаштований так, щоб він був ідентичний імені користувача (наприклад, admin:admin). Це здавалося розумним вибором, оскільки ідентичні комбінації імені користувача та пароля зазвичай використовуються в атаках методом перебору.

Після менш ніж 24 годин виконання експерименту було виявлено, що EC2 примірник сервера приманки повністю зазнав краху. Збій, мабуть, був викликаний незвичайною поведінкою CRON: системні журнали показали, що він рекурсивно перезапускав себе до тих пір, поки не було занадто багато процесів для хоста EC2, щоб обробити його, що в кінцевому підсумку призвело до аварійного завершення примірника.

Після перевірки журналів, згенерованих контейнером маршрутизатора, було зареєстровано кілька успішних сеансів підключення для SSH і telnet. Однак після вивчення файлу історії *.bash* жодна з виконаних команд не виглядала такою, що могла б викликати таку поведінку. Таким чином, був зроблений висновок, що аварійне завершення сталося через перевантаження ресурсів системи.

Хоча система перебувала в автономному режимі досить довго, перш ніж була виявлена помилка, за кілька годин, протягом яких проводився

експеримент, було зафіксовано кілька успішних спроб входу в контейнер маршрутизатора. Однак, все ще не було атак, які досягли мережі приманок. Вміст файлу історії *.bash* показав, що за час проведення експерименту були виконані тільки 6 команд. Три аналогічні послідовності команд були виконані наступним чином:

cat /proc/mounts; (/bin/busybox LWQTL —— :)

cd /dev/shm; cat .s —— cp /bin/echo .s; (/bin/busybox LWQTL —— :)

cat /proc/mounts; (/bin/busybox GQZJN —— :)

cd /dev/shm; cat .s —— cp /bin/echo .s; (/bin/busybox GQZJN —— :)

cat /proc/mounts; (/bin/busybox WEGOD —— :)

cd /dev/shm; cat .s —— cp /bin/echo .s; (/bin/busybox WEGOD —— :)

Слід відзначити той факт, що ці команди намагалися викликати оболонку BusyBox, яка не була встановлена в контейнері маршрутизатора. У своєму аналізі поведінки IoT ботнету Hajime Едвардс та ін. [54] зазначили майже ідентичні виклики оболонки BusyBox, призначені для визначення властивостей системи жертви через механізм зняття цифрових відбитків. Реальна система, в якій встановлено BusyBox, буде повертати відповідь «CHDGL: applet not found». Ці дослідники також відзначили призначення послідовності команд cat/proc/mounts, яка «перевіряє монтування системи на наявність місця для запису в цільовій файловій системі», і послідовності *cd/var; cat .s - cp/bin/ echo .s;/Bin/busybox ECCHI*, яка серед іншого «вибере перший доступний для запису шлях, який не є */proc*, */sys*, або */*, і використовує його як робочий шлях». Той факт, що ці команди були дотримані, може вказувати на причетність ботнету Hajime.

В цілому, результати цієї ітерації експерименту були цікавими, але не повними, оскільки жоден зловмисник не просунувся далі початкової стадії. Тому було вирішено запустити цей ітерацію в третій раз, вже з встановленою утилітою.

4.2.4 Експеримент 1, Спроба 3

Після додавання утиліти BusyBox в контейнер маршрутизатора, експеримент був розгорнутий в третій раз. Додаткових налаштувань для цієї утиліти не було потрібно.

Як і планувалося, ітерація експерименту проводилась протягом двох днів, перш ніж систему було переведено в автономний режим і розглянуто результати. На цей раз мережеві події в syslog показали, що ще декільком зловмисникам вдалося успішно пройти аутентифікацію в контейнері маршрутизатора як по SSH, так і по telnet.

Результати були наступними:

1) Навіть після повторення цього експерименту в третій раз, жоден зловмисник не пройшов від контейнера маршрутизатора до мережі приманок Cowrie. Це стало очевидним з того факту, що на панелі управління Kibana, яка розміщена на сервері управління, не було жодних візуалізацій. У файлі історії *.bash* також не було виявлено команд, які вказували б на те, що зловмисник досліджував мережу dmz.

2) Як і в попередній ітерації, у файлі історії *.bash* були виявлені команди, які можна віднести до зловмисних. Однак спостерігалася тільки одна послідовність, аналогічна виділеній в попередньому розділі. Замість цього неодноразово спостерігалася інша послідовність: *enable; system; shell; sh*. Це була ще одна послідовність, яка обговорювана Едвардсом і ін., і яку вони пояснюють як послідовність, «яка відправляється в сліпий спробі перейти на будь-який специфічний для виробника інтерфейс командного рядка (CLI), який реалізує сервер Telnet... Якщо будь-яка команда завершиться невдало, вона зазнає невдачі» [54]. При спробі вручну виконати ту ж послідовність всередині контейнера маршрутизатора було виявлено, що системні команди та команди оболонки не виконувалися всередині контейнера, що і пояснює той факт, що послідовність атаки завершилася в цей момент.

3) Виявилося, що один зловмисник все-таки отримав доступ до системи. Розглянута історія *.bash* показала, що після встановлення утиліти *wget* зловмисник завантажив Python-скрипт. Після відвідування посилання, через яке відбулось завантаження, був виявлений сценарій, що містить інформацію про ліцензування відкритого вихідного коду. Веб-сторінка, з якої відбулось завантаження, зображена на рисунку 4.1.

Рисунок 4.1 – Веб-сайт, з якого відбулось завантаження

Сценарій був простежений до Github проекту, який називається speedtest-cli [96]. Це інструмент, який використовується для перевірки пропускної здатності мережі, доступної для пристрою, що, безсумнівно, становить інтерес для зловмисника, який мав намір провести об'ємну мережеву атаку, таку як DDoS. Інші дії в тому ж сеансі включали створення нового користувача *huawei*, перевірку працездатності системи за допомогою команди *uptime* і перевірку специфікацій системи за допомогою команди *lscpu*.

4.2.5 Завершення експериментів з приманкою

З проведених ітерацій стало зрозуміло, що для експериментів з приманками є характерний високий ступінь непередбачуваності. З огляду на те, що для завершення всіх 12 ітерацій запланованих експериментів виділено 4 тижні, було вкрай важливо, щоб протягом цього часу виникали лише мінімальні затримки у часі.

Через проблеми, описані в розділі 4.2, перша з 12 ітерацій була розгорнута три рази. Опираюсь на це було зроблено висновок, що проведення решти ітерацій є неможливим, враховуючи, що для цього залишалося менш як три тижні. Таким чином, проведення запланованих експериментів було припинено.

4.3 Оцінка досягнень і обмежень дослідження

Досягнення, і обмеження дослідження оцінюються та обговорюються в цьому розділі.

4.3.1. Моніторинг кібер-інцидентів, керований Honeypot

Перша мета цього дослідження полягала в тому, щоб розробити монітор кіберінцидентів, керований приманкою, який можна було б розгорнути в сучасній IT-інфраструктурі, щоб організації мали змогу реалізувати активний захист мережі.

Як важливий компонент оцінки та одна з нових особливостей впровадженої системи моніторингу інцидентів, яка запропонована в цьому дослідженні, є ефективність розгортання мережі Honeynet.

В корисному наборі критеріїв для оцінки їх розгортання в мережі Honeynet, який викладено В. Чіном та ін., [97] та використовується В. Пісарчіком та ін., [61] вказується ряд ключових критеріїв, яким повинна відповідати ефективна мережа Honeynet. Їх опис стосовно до реалізованої системи наведено нижче:

1) Масштабованість

В ході дослідження було випробувано найбільше розміщення Honeynet: 11 контейнерів з приманками, які розміщувалися безперервно протягом 2 днів без будь-яких збоїв. Однак в цей період взаємодія з системою була дуже

незначною, що означало обмежене навантаження на системні ресурси (що не є характерним для виробничого середовища).

Теоретично, мережа Honeynet може підтримувати велику кількість контейнерів приманок. Можливість додавати більше приманок багато в чому залежить від доступних системних ресурсів: хост-система повинна мати достатньо ресурсів для підтримки необхідної кількості розгорнутих контейнерів. Проте, їх все ж буде менше, ніж потрібно для еквівалентної кількості віртуальних машин на одному хості.

2) Гнучкість

Запроваджене рішення Honeynet є гнучким у своїй працездатності, конфігурації та обслуговуванні:

- Контроль над Honeynet є простим з адміністративної точки зору: управління контейнерами може здійснюватися через хост навіть під час їх роботи, що означає, що налаштування мережі та адміністративний доступ до ресурсів контейнерів можуть бути виконані «на льоту» в міру потреби;
- Використання приманки Cowrie також сприяє гнучкості Honeynet, оскільки її налаштування дозволяє протистояти навіть невідомим раніше загрозам;
- Відносно простим є також використання будь-яких інших приманок: для альтернативних приманок можуть бути визначені нові Docker-файли, які згодом інтегруються в систему за допомогою простого редагування сценарій розгортання без необхідності вносити будь-які зміни в конфігурацію мережі.

3) Стримування атак

Під стримуванням атак розуміється обмеження поширення атак, якщо приманка була скомпрометована. Хоча стримування атаки ніколи не забезпечується повністю, заходи, прийняті для забезпечення стримування атаки, описані в Розділі 2.2.4, пом'якшують небажане поширення атаки за межами Honeynet:

- Ізоляція хоста від мережі dmz, в якій розміщується Honeynet, обмежує здатність зловмисника запускати мережеву атаку зсередини Honeynet.
- Ретельно продумані обмеження, що накладаються на зловмисника всередині приманок Cowrie, означають, що вкрай малоймовірно, що зловмисник зможе почати атаку з цих приманок. Той факт, що Cowrie є емульованим середовищем, зокрема, обмежує дії зловмисників, оскільки вони не взаємодіють з реальною системою з мережевим стеком, ОС і потужними утилітами.
- Контейнер маршрутизатора з високим рівнем взаємодії є єдиним компонентом системи, в якому захист від атак може бути не повністю забезпечений, оскільки цей контейнер взаємодіє безпосередньо з мережею, до якої підключений хост. Це потенційна область для поліпшення, у якій може бути досягнуто підвищений рівень ізоляції, якщо приділити увагу дослідженню використання альтернативних мереж, які не взаємодіють з хост-системою.

4) Прихованість

Прихованість відноситься до здатності приманки обдурити зловмисника і переконати його в тому, що це законна система. Важко дати однозначний висновок щодо цієї властивості розгорнутої honeynet, враховуючи, що взаємодії зловмисників з системою в кращому випадку були обмежені.

Безумовно, існує можливість поліпшити прихованість Honeynet, ґрунтуючись на невеликих даних, які були отримані в ході експериментів, описаних в розділі 4.2, оскільки на кожній ітерації було встановлено, що атакуючі IoT пристрої ботнети не просувають свої атаки за межі своїх відомих початкових фаз [20];[54]. Щоб зробити будь-які остаточні висновки про цю властивість, буде потрібно значно більше часу та експериментів. Проте, властивості контейнерної мережі Honeynet піддаються оманливим діям зловмисників щодо легітимності середовища, оскільки всі примірники розміщуються в контейнерах з реальними образами ОС і мережевими стеками. В. Чін та ін. припустити, що прихованість може бути досягнута за допомогою

таких заходів, як фальсифікація високої затримки мережі [97]. Зловмисник, який перевіряє легітимність цієї мережі на основі таких властивостей, отримає законно затримані відповіді через використання контейнерів в системі.

5) Управління ресурсами

Цей критерій відноситься до наявності простих механізмів розподілу ресурсів в мережі Honeynet, що значною мірою забезпечується шляхом використання контейнерної екосистеми Docker.

- Природа контейнерів полягає в тому, що ресурси, які виділяються для запуску програми, суворо контролюються і повністю визначаються за допомогою образу. Для того, щоб виділити додаткові ресурси, потрібно просто оновити його значення.
- Ресурси системи хоста, які споживаються контейнерами, управляються, головним чином, за допомогою специфікації аргументів при створенні контейнерів. Як зазначено в документації Docker, «за замовчуванням контейнер не має обмежень по ресурсах... Docker надає способи управління обсягом пам'яті, ЦП або пристроями введення-виведення, які контейнер може використовувати, встановлюючи прапори конфігурації часу виконання команди *docker run*» [98]. Хоча ці обмеження на ресурси не були явно враховані як частина розробки контейнерної Honeynet, їх було б досить просто додати сценарій розгортання.

6) Простота розгортання

В. Чін та ін. стверджують, що для того, щоб мережа Honeynet вважалась простою в розгортанні, «користувачі повинні бути не обтяжені складними конфігураціями та процедурами для налаштування своїх приманок».

У запропонованій системі використання контейнерів та автоматизація конфігурації мережі за допомогою сценаріїв означає, що повністю сконфігурована і налаштована мережа підключених контейнерів може бути видалена і повторно розгорнута на хості Linux приблизно за 5 хвилин.

Важливість простоти використання також підкреслюється В. Чіном та ін. [97] Засоби, реалізовані в сценарії розгортання *do_honeypot.sh*, усувають

додаткову складність процесу розгортання, надаючи прості консольні повідомлення для керівництва при розгортанні та налаштування системи.

Таким чином, робиться висновок про те, що, хоча можливості для поліпшення різних аспектів реалізованої мережі Honeynet й існують, проте, вимоги, викладені в цій класифікації, значною мірою були дотримані.

Обмеження, яке було встановлено при оцінці архітектури системи, полягає в централізації обробки журналів в моніторі інцидентів. Хоча централізована агрегація даних приманки для візуалізації в системі забезпечує кореляцію даних атаки для забезпечення цілісного уявлення всієї мережі Honeynet, це також означає, що система спроектована таким чином, що у неї є єдина точка відмови.

Поточна архітектура системи неідеальна з точки зору надійності системи, оскільки, якщо EC2 примірник сервера управління буде перевантажений і аварійно завершиться, система моніторингу інцидентів більше не зможе надавати визначену послугу.

Однак є деякі особливості архітектури впровадженої системи, які роблять її більш масштабованою: зокрема, те, що Honeynet в цьому розгортанні розміщується локально з Docker, а не як розподілена мережа. При оцінці монітора інцидентів TraCINg, E. Васіломанолакіс та ін. [43], відзначають, що їх розподілена архітектура створює вузьку область щодо конкуренції за ресурси «коли кількість нових датчиків значно збільшується», оскільки кілька розподілених приманок окремо надають свої результати в централізовану систему моніторингу інцидентів. В розробленій системі дані за допомогою томів Docker і CRON агрегуються локально, а не передаються кожною приманкою, що зменшує розмір мережевого трафіку до примірника управління.

Використання візуалізації в цьому проекті було обумовлено необхідністю зробити рішення на основі приманки більш зручними, надаючи адміністраторам можливість легко отримати цілісне уявлення про стан активного захисту мережі у своїх системах.

Хоча зібрати експериментальні дані приманками Cowrie, чиї журнали можна було обробляти та візуалізувати через веб-інтерфейс Kibana, так і не вдалось через обмеження в часі, були проведені деякі симульовані атаки, що дозволяють продемонструвати візуалізацію для оцінки.

На рисунку 4.2 зображена панель візуалізації Kibana, яку було створено в рамках рішення моніторингу інцидентів. Наведено кілька діаграм, які вимірюють все: від популярності паролів на приманки до частки telnet і SSH-атак, які кожна приманка Cowrie отримала. Короткий опис стану загроз в мережі Honeynet міститься в панелі інструментів.

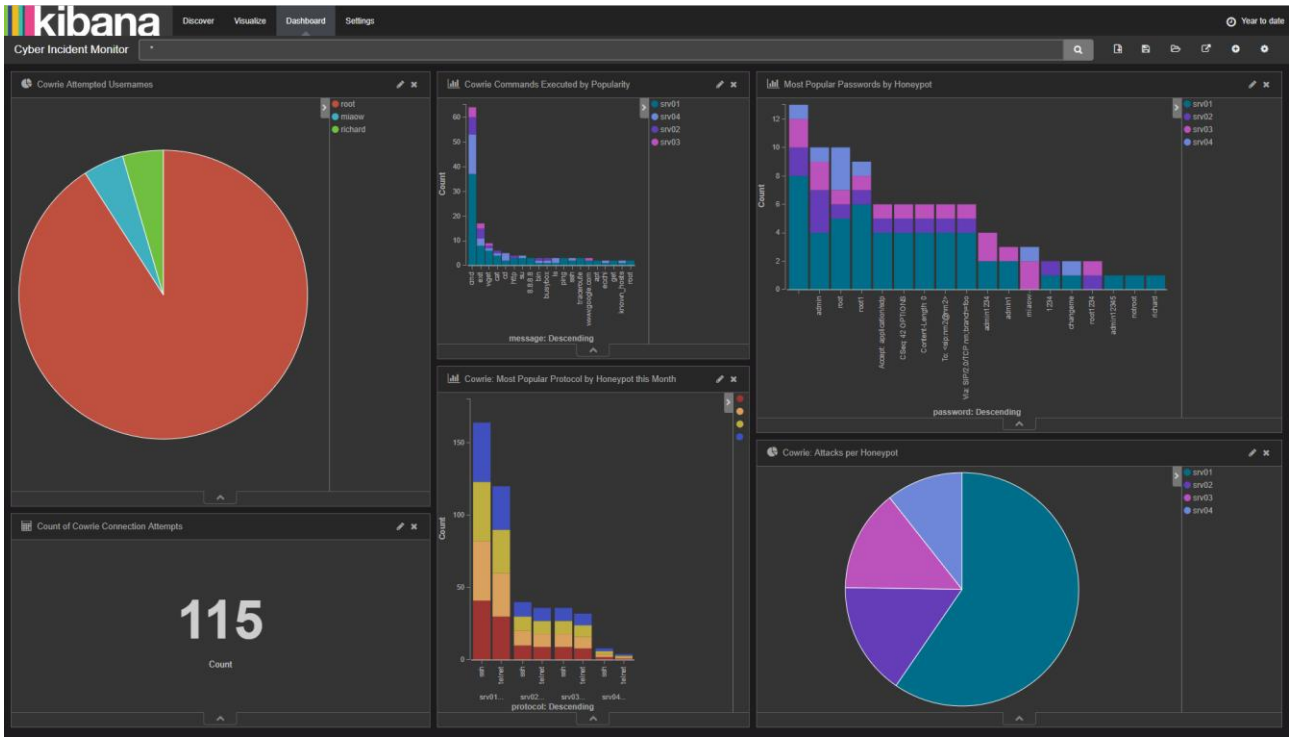

Рисунок 4.2 – Моніторинг кіберінцидентів на за допомогою засобу моніторингу Kibana

Візуалізація частки атак, отриманих кожною приманкою Cowrie, зображена на рисунку 4.3:

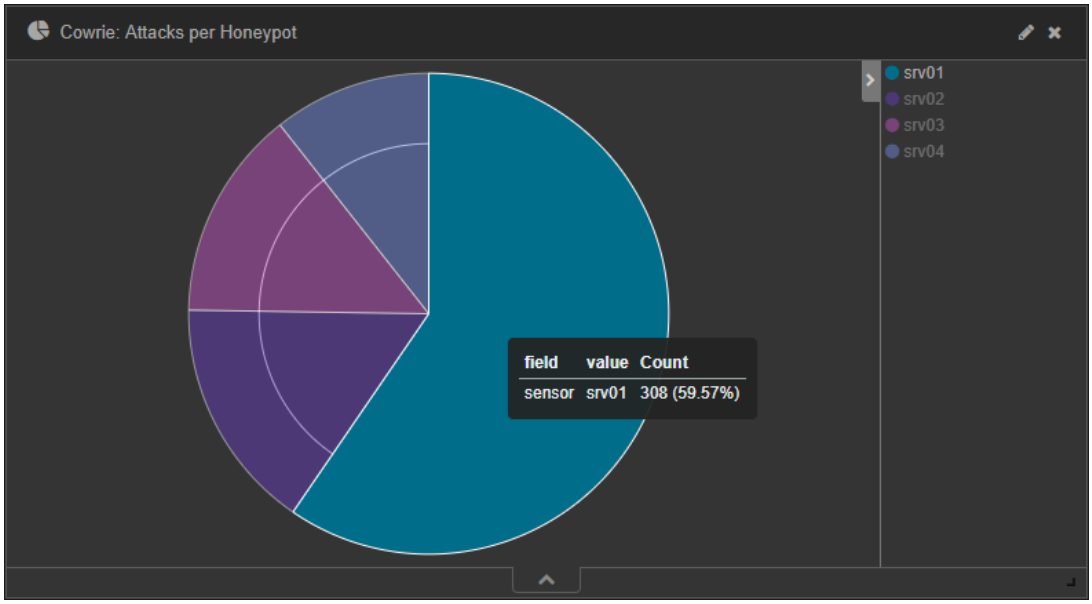

Рисунок 4.3 – Кругова діаграми частки атак, яку отримала кожна приманка Cowrie

Візуалізація частки атак SSH і Telnet, отриманих кожною з приманок Cowrie кожного тижня протягом місяця, зображена на рисунку 4.4:

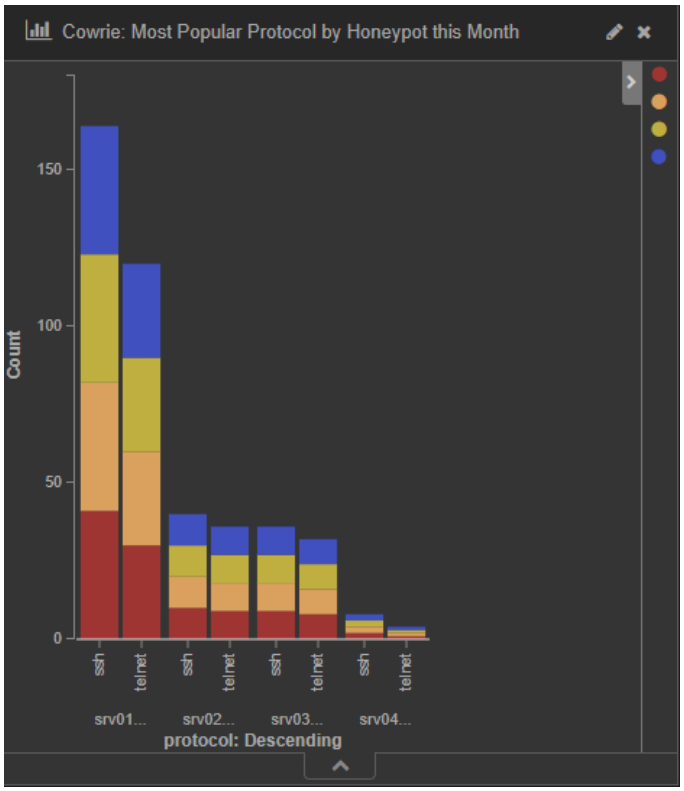

Рисунок 4.4 – Візуалізація найпопулярніших протоколів на приманках за місяць

Що можна відзначити відносно цих візуалізацій, так це ефективність, з якою інформація передається користувачеві: замість самостійного аналізу величезної кількості даних журналу приманки, корельована інформація про поведінку і моделі атак вже коротко представлена в візуалізації. Той факт, що журнали, які використовуються для їх створення, відправляються з сервера приманки приблизно кожні 60 секунд, означає, що адміністратору набагато легше зрозуміти поточні загрози для його системи.

4.3.2 Проектування адаптивних приманок

Передбачалося, що в цьому розділі буде проведена кількісна оцінка, однак через проблеми, що виникли при виконанні всіх запланованих ітерацій експериментів, оцінка внеску цього дослідження в розробку адаптивних приманок обмежена неповними результатами, отриманими в Розділі 4.2

Як було встановлено, процес проектування приманок для адаптації до динамічних загроз займає багато часу. Непередбачені проблеми, з якими зіткнулися при її ефективної реалізації, свідчать про непередбачуваність розробки адаптивних приманок в цілому.

Той факт, що в експериментах, описаних в розділі 4.1, були враховані внески тільки трьох характеристик приманок, пояснюється обмеженим часом, протягом якого експерименти могли проводитись.

Порівнюючи час, який витрачається на проведення аналогічних дослідницьких експериментів в тісно пов'язаних проектах, які обговорювались в розділі 1.4, результати були наступними:

- Дослідники, які беруть участь в проекті IoTPot, провели експерименти для оцінки ефективності у 2 окремих етапи: 144-денний випробувальний період з 2014/11/07 по 31/03/2015, який використовувався для «розуміння поведінки зловмисників», і «обговорень правильного розташування приманок», після якого було організовано 28-денний стабільний період з

01/01/2015 по 09/05/2015 під час якого проводилися більш структуровані експерименти [58].

- У своїй оцінці продуктивності приманки в моніторі інцидентів TraCINg Е. Васіломанолакіс та ін. пояснюють, що для отримання значущих результатів їм знадобилось п'ятимісячне розгортання [43].

Ці результати додатково ілюструють непередбачуваність оцінки ефективності приманок. Можливо, це упущення, що час, необхідний для проведення таких експериментів, не був реалізований на більш ранній стадії.

Визнаючи той факт, що жоден з проведених експериментів не може надати переконливих доказів щодо конструкції адаптивних приманок, необхідно було розглянути те, що було необхідним (1), достатнім (2) і додатковим (3) для досягнення початкового завдання.

1) Необхідне

Необхідно отримати знання про внесок конструкції honeypot в її ефективність в залученні атак за допомогою експериментів.

2) Достатнє

Досить того, що отримані знання не є остаточними, але засновані на експериментальних даних.

3) Додаткове

Крім того, отримані на основі експериментальних даних знання є остаточними та переконливими.

Зрозуміло, що отримання переконливих результатів, які стосуються ефективного проектування адаптивних приманок, стало неможливим через брак часу, що в кінцевому підсумку призвело до дострокового припинення фази експерименту. Однак дані, отримані в результаті трьох спроб, хоча і не були остаточними, але дали деяке уявлення про природу проектування адаптивних приманок.

- Наявність очікуваних утиліт та наборів інструментів в середовищі приманки, безумовно, має вирішальне значення для успіху в заохоченні атак. Це було видно з даних, зібраних в спробах 2 і 3 проведеної ітерації

експерименту, коли зловмисникам не вдавалось скомпрометувати систему через відсутність деяких утиліт.

- Підтримка протоколів і сервісів, які зазвичай використовуються зловмисниками, є важливою особливістю ефективних приманок: це очевидно завдяки величезній кількості отриманих атак методом перебору.
- Використання структури приманки на основі поточних знань про атаки є корисною відправною точкою для вивчення атак. Ймовірно, що експерименти були б більш успішними, якби при розробці приманки маршрутизатора з високим ступенем взаємодії більше уваги приділялося налаштуванню середовища для надання відповідей і утиліт, які шукають активні ботнети IoT, знання про які вже існують в дослідженнях [20];[54].
- Приманки повинні бути призначені для боротьби як з автоматичними, так і з людськими атакуючими. На підставі результатів Т Баррона та ін. не очікувалося, що в середовищі приманки будуть зустрічатись атаки з боку зловмисників-людей [49].

Той факт, що один сеанс атаки таким зловмисником був проведений під час експериментального періоду показує, що приманки повинні бути спроектовані так, щоб адаптуватися до загроз з боку людей, а не лише до більш передбачуваних моделей атак ботів.

Як доведено під час проведення експериментів в розділі 4.2, той факт, що на контейнері маршрутизатора не був доступний кейлогінг, серйозно обмежує висновки, які можна зробити щодо будь-яких отриманих експериментальних результатів. Наприклад, хоча в історії *.bash* спостерігалися шаблони виконання команд, відсутність міток часу та ідентифікаторів сеансів для цих подій означало, що були зроблені висновки про введення цих послідовностей від одного зловмисника.

Цілком ймовірно, що деякі з моделей атак, які описано в розділі 4.2, могли б бути іншими, якби для розміщення системи використовувалося інше рішення для хостингу. Як відзначають Е. Васіломанолакіс та ін. у своїй реалізації монітора інцидентів для приманок, [43] багато постачальників

хмарних послуг публікують свої діапазони IP-адрес в загальнодоступному Інтернеті. AWS є одним них [99]. У своїй статті, присвяченій вивченню відмінностей між людськими та автоматизованими атаками, Т. Баррон та ін. також відзначає, що «місце розташування і хост мають значення... якщо хтось працює з обмеженим бюджетом і хоче максимізувати кількість зібраних (методом перебору) IP-адрес, то AWS, мабуть, є тією інфраструктурою, яка буде цьому сприяти» [49].

Оскільки AWS є дуже відомим і популярним провайдером хмарного хостингу, він, ймовірно, є привабливою метою для зловмисників, які знають діапазони його загальнодоступних IP-адрес. Таким чином, існує ймовірність деякої упередженості в результатах, отриманих в розгортанні, подібному тому, яке було реалізовано в цьому дослідженні, коли всі приманки розміщуються в одному географічному регіоні одним постачальником послуг хостингу.

Цікавим є спостереження, яке стосується зняття відбитків системи в середовищах приманок. Воно полягає в тому, що зловмисник має багато можливостей визначити, що він знаходиться всередині контейнера, якщо приманка дійсно є контейнером. Це було відзначено під час впровадження системи після виконання команди *cat/proc/1/cgroup*, яка дала відповідь, зображену на рисунку 4.5. Консольний вивід явно містить в собі кілька згадувань «docker», які будь-який зловмисник, обізнаний про Docker, відразу розпізнає як контейнерну середу.

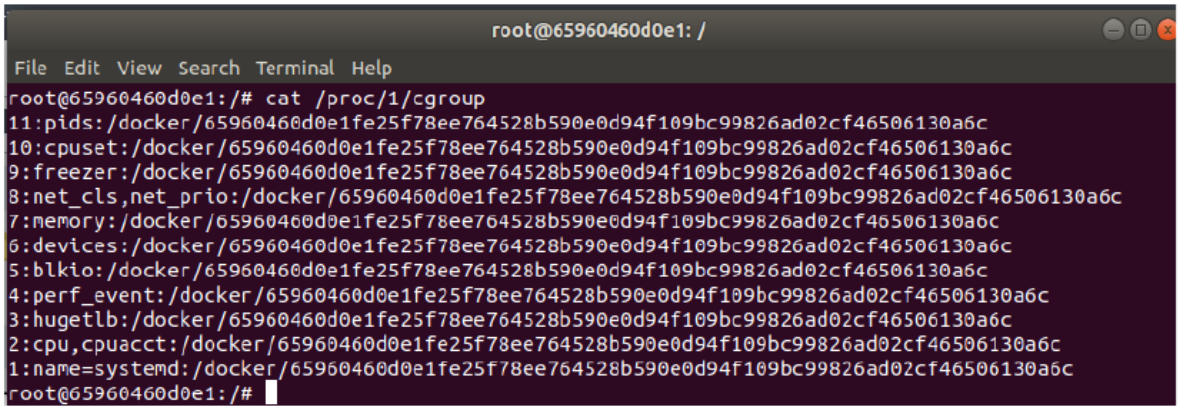

Рисунок 4.5 – Зняття відбитків системи в середовищі Docker

Це приводить до таких самих висновків, як і визначених А. Кедровіч та ін. в дослідженні схильності контейнерів до зняття відбитків системи [62]. Однак, оскільки організації все частіше включають контейнери у свою інфраструктуру, ця інформації, швидше за все, не змусить ідентифікувати контейнери як потенційну приманку. Проте, безумовно, варто враховувати, що, якщо контейнери для приманок починають використовуватися більш широко, зняття відбитків цих систем як середовища для приманок також може стати більш поширеним.

4.4 Висновок до розділу

Зрозуміло, що непередбачуваність проведення експериментів з приманкою стала основною перешкодою для отримання вимірних даних і надання остаточних висновків щодо конструкції адаптивних приманок.

Однак, непереконливі, корисні ідеї з проведених експериментів вже ж були отримані. Це є обнадійливим показником потенційної можливості внесення реального вкладу в експерименти такого роду і забезпечує міцну основу для можливого завершення ітерацій проекту в майбутньому.

5 ОБГРУНТУВАННЯ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ

5.1 Загальний підхід до визначення економічної ефективності розробки

Обов’язковою складовою частиною будь-якого інжинірингового проекту є фінансові витрати на різних етапах виконання робіт. Відповідно, важливо вірно здійснити фінансову оцінку передбачуваних витрат, продуктивність, корисність та, в результаті, економічну ефективність проекту.

Наукомісткі розробки та дослідження, на відміну від корпоративних, не завжди супроводжують за мету отримання прибутку або ж іншої матеріальної вигоди. В багатьох випадках, проекти наукового спрямування не є економічно вигідними. Однак, вони є рушійною силою прогресу, дослідженням незвіданих галузей та проблем, які, у свою чергу, майже завжди впливають на майбутні різнопланові розробки. Тому дуже часто наукова діяльність стимулюється зовнішніми інвестиціями та підтримкою держаних інститутів, міжнародних грантів. В плані використання результатів досліджень та на основі отриманих моделей можна робити прогнози по впровадженню нових методик та принципів організації роботи діагностичних установ. Різноманітні медичні центри зможуть скористатися на практиці розробленою системою для покращення існуючих та отримання нових діагностичних методик.

Оцінка вартості дослідницьких розробок базується на витратному підході: використанні первісної вартості об'єктів, виходячи з фактичних витрат на розробку та доведення до комерційного використання з урахуванням амортизації. Так, як результати роботи у вигляді математичних моделей та реалізованого на їх основі ПЗ не буде використовуватися в комерційних цілях та не підлягатиме продажу, а становить наукову та інтелектуальну цінність, то доходів від продажу ПЗ та розробки як такого не передбачається. Іншими словами, всі вкладені кошти та витрати на розробку даного рішення є не взаємоокупними, що несуть лише витрати у кількості залучених ресурсів та матеріальних засобів.

Згідно з Статтею 8 Закону № 3792-12 передбачено, що твори наукового характеру та комп'ютерні програми є об'єктами авторського права [100]. У разі реєстрації виконаної роботи як інтелектуальної власності та авторського права у державній службі інтелектуальної власності України необхідно буде сплатити збори за державну реєстрацію (382,5 грн).

Для отримання відмінних результатів експериментів та доцільності розробки такого спеціалізованого ПЗ потрібні відповідні затрати на дослідження та розробку. Це і становитиме основу витрат, які будуть здійснені протягом підготовки та виконання реалізації даного рішення. Уявно модель витрат можна поділити на дві основні частини: витрати, пов'язані на дослідження предметної області, побудову математичних моделей, отримання попередніх результатів експериментів, та частину реалізації програмної системи, архітектури та тестування.

Враховуючи залежність якості кінцевого продукту від кваліфікації програмістів, потрібно сконцентрувати увагу на якості та результативності розробки. Для цього потрібно провести ґрунтовний аналіз предметної області, залучити найновіші та інноваційні технології, провести ґрунтовне тестування та оцінку результатів розробки. Для забезпечення хорошої результативності при розробці доводиться йти на додаткові заходи заохочення та стимулювання у вигляді преміювання працівників, підтримання наукового дослідження досвідом іноземних науковців, дорогоцінних лабораторних дослідів.

До створення ПЗ можуть бути залучені позаштатні програмісти як зареєстровані, так і не зареєстровані підприємцями. В обох випадках співпраця з ними здійснюється на підставі цивільно-правового договору, найчастіше – договорами підряду. Щодо оподаткування виплат за договором підряду, то все залежить від того, чи зареєстрований виконавець підприємцем. Важливим етапом розробки ПЗ є його тестування, яке виконується тестувальником за певну винагороду. Якщо тестувальники не перебувають з підприємством у трудових відносинах, оплата виконується на основі договору підряду на

виконання робіт з тестування ПЗ. Сам же результат розробки не оподатковується, адже не є комерційним проектом і не спрямований на продаж.

5.2 Розрахунок витрат на розробку мережі Honeynet

Для оцінки економічного ефекту від впровадження будь-якого інноваційного продукту необхідно розрахувати витрати на його придбання, або розробку. Витрати на проектування мережі за моделлю honeynet розраховуються шляхом складання кошторису на проектування. Роботи з проектування мережі за моделлю honeynet на підприємстві виконуються проектною групою, склад якої представлено в таблиці 5.1:

Таблиця 5.1 – Штатний розклад проектної групи

Категорія робітників	Кількість робітників, чол.	Посадовий оклад, грн/міс.
Головний інженер-проектувальник	1	14000
Інженер-проектувальник	1	10000
Разом	2	17000

Проектування мережі за моделлю honeynet включає кілька основних етапів, склад яких представлений в таблиці 5.2:

Таблиця 5.2 – Перелік основних етапів проектування

Етап	Зміст робіт етапу	Кількість виконавців, чол.	Посада	Тривалість роботи, днів
Підготовчий	Ознайомлення із завданням на проект	1	Головний інженер-проектувальник	1
	Підбір і вивчення технічної літератури	1	Інженер-проектувальник	1
Аналіз вимог	Оформлення технічного завдання на проектування	1	Головний інженер-проектувальник	2

Продовження таблиці 5.2

Технічний проект	Підбір і розрахунок обладнання і ПЗ	1	Інженер-проектувальник	3
	Випуск робочого проекту	1	Інженер-проектувальник	3

Розрахунок витрат на заробітну плату проектної групи представлений в таблиці 5.3:

Таблиця 5.3 – Розрахунок витрат на заробітну плату проектної групи

Посада	Оклад, грн/міс	Оплата, грн/день	Тривалість робіт, дні	Разом, грн.
Головний інженер-проектувальник	14000	666	3	2000
Інженер-проектувальник	9000	428	7	3000
Разом по тарифу				5000
Доплати (20% від тарифу)				1000
Основна заробітна плата				6000
Додаткова заробітна плата (10% від основної)				600
Єдиний соціальний внесок (22%)				1452
Разом				5148

Перелік і розрахунок вартості необхідного ПО для побудови мережі за моделлю honeynet представлений в таблиці 5.4:

Таблиця 5.4 – Перелік і розрахунок вартості необхідного ПО і устаткування

Назва ПЗ	Одиниця вимірювання	Кількість	Ціна. грн	Вартість, грн
Cowrie	шт.	1	0	0
Docker	шт.	1	0	0
ELK	шт.	1	0	0
AWS EC2 для приманок	шт.	1	3,9 грн в годину	34164 грн на рік
AWS EC2 для управління	шт.	1	1,2 грн в годину	10512 грн на рік
Разом				44676

Отже, сумарні витрати на розробку та впровадження мережі складають:

5148 + 44676 = 49824 грн

Крім витрат на розробку і впровадження мережі Honeynet існують експлуатаційні витрати, які являють собою поточні витрати, пов'язані з експлуатацією проектованої мережі. Експлуатаційні витрати включають такі витрати, як оплата праці обслуговуючого персоналу, витрати на матеріальні та запасні частини, електроенергію та ін. Мережа Honeynet не вимагає додаткових експлуатаційних витрат, оскільки вона буде обслуговувати той самий персонал, що і реальну мережу підприємства, а додаткових витрат на матеріали не потрібно. Тому експлуатаційні витрати враховуватись не будуть.

5.3 Розрахунок економічної ефективності проектованої мережі

Побудова обчислювальної мережі Honeynet відноситься до заходів по захисту інформації на підприємстві.

На практиці у сфері інформаційної безпеки використовується оцінка економічної ефективності, яка заснована на суб'єктивній точці зору. Парадокс ситуації полягає в тому, що IT- і ІБ-фахівці прекрасно розуміють значення і важливість реалізації заходів, спрямованих на підвищення рівня інформаційної безпеки, а для об'єктивної оцінки економічного ефекту немає універсальних методів. Економічний ефект являє собою перевищення вартісних оцінок кінцевих результатів над сукупними витратами ресурсів (трудових, матеріальних і т.п.) за розрахунковий період.

Витрати на забезпечення інформаційної безпеки слід вважати ефективними, якщо вони забезпечують виконання вимог нормативних документів і стандартів, прийнятих державою, а також концепції інформаційної безпеки організації.

Кінцевим результатом впровадження та проведення заходів щодо забезпечення інформаційної безпеки є значення запобіглих втрат ($З_{вт}$), яке розраховують, виходячи з ймовірності виникнення інциденту інформаційної

безпеки та можливих економічних втрат від нього до і після реалізації заходів з забезпечення інформаційної безпеки на об'єкті:

$$З_{вт} = З_1 – З_2 (1), \quad (5.1)$$

де $З_1$ – втрати від реалізації загроз до впровадження заходів, що підвищують рівень інформаційної безпеки;

$З_2$ – втрати від реалізації загроз після впровадження заходів, що підвищують рівень інформаційної безпеки.

По суті, $З_{вт}$ є різницею втрат до і після реалізації заходів, спрямованих на підвищення рівня інформаційної безпеки, і в цілому зображає ту частину прибутку, яка могла бути втрачена.

Показником, який безпосередньо впливає на ефективність роботи підрозділу по захисту інформації, є витрати на його утримання (У) за розрахунковий період, до яких входять оплата праці фахівців, закупівля та утримання технічних засобів захисту інформації та ін.

Коефіцієнт ефективності підприємства по захисту інформації за розрахунковий період повинен об'єктивно і доступно відображати суть його діяльності. Оскільки від ефективності підрозділу по захисту інформації безпосередньо залежить кількість інцидентів інформаційної безпеки, то коефіцієнт ефективності можна виразити як різницю явної повної ефективності та відносної частоти виникнення інцидентів інформаційної безпеки:

$$К_{еф} = 1 – I_{еф.р.п.}/I_{max\ р.п} \quad (5.2)$$

де $I_{еф.р.п.}$ – кількість інцидентів інформаційної безпеки за розрахунковий період;

$I_{max.р.п.}$ – максимально можлива кількість інцидентів інформаційної безпеки за розрахунковий період.

При визначенні максимально можливої кількості інцидентів інформаційної безпеки ($I_{max.p.n.}$) необхідно користуватися статистичними даними. Якщо ж вони недоступні, можна скористатися методом експертно-аналітичної оцінки.

Коефіцієнт ефективності організації із захисту інформації завжди приймає значення $0 \leq K_{еф} \leq 1$.

Таким чином, економічна ефективність ($E_{еф}$) організації із захисту інформації за розрахунковий період може бути визначена за формулою:

$$E_{еф} = (З_{вт} - У) * K_{еф} \quad (5.3)$$

Орієнтовна структура наслідків від нереалізації захисних заходів захисту інформації в сучасних організаціях представлена на рисунку 5.1:

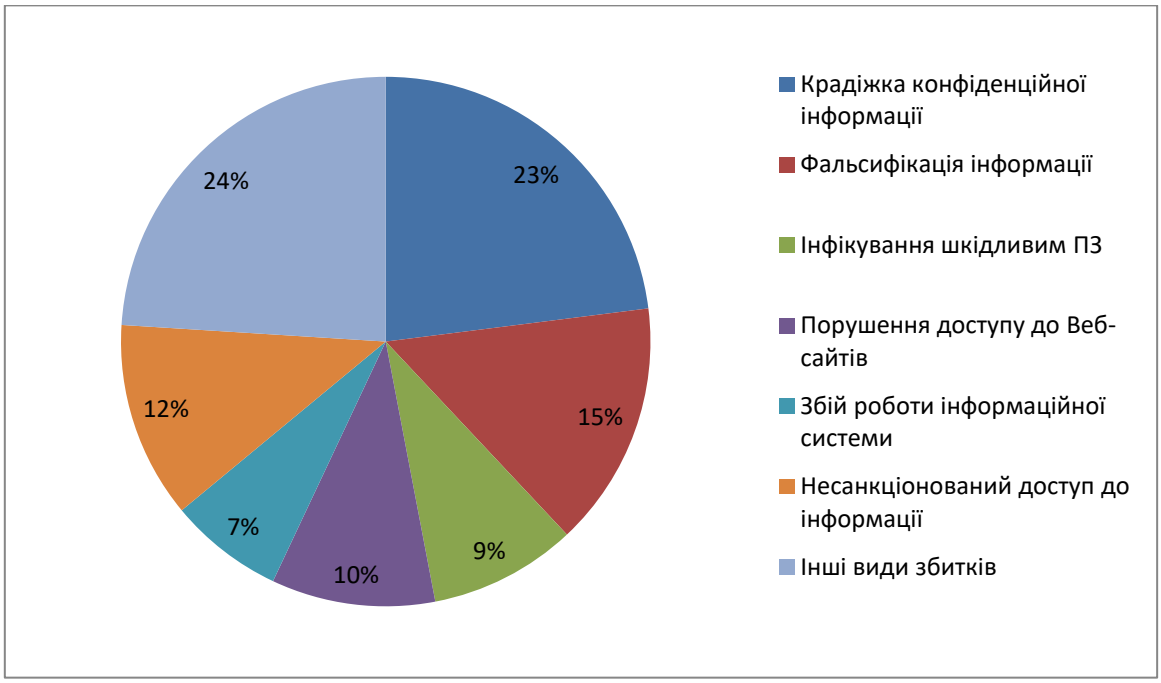

Рисунок 5.1 – Структура наслідків від нереалізації захисних заходів захисту інформації в сучасних організаціях

Сумарні витрати на створення мережі Honeynet складають 49824 грн (з них 44676 грн є щорічною витратою).

Обсяг середньорічних втрат компанії через інциденти інформаційної безпеки становить 190000 грн.

Значення запобіглих втрат:

$$З_{вт} = 190000 – 190000\ (1 – 0,23 – 0,24 – 0,12 – 0,07 – 0,1) = 144400$$

Коефіцієнт ефективності підприємства щодо захисту інформації при створенні мережі Honeynet становить КЕФ = 0,96.

Економічна ефективність щодо захисту інформації на підприємстві за рік становить:

$$Е_{еф} = (З_{вт} - У) * К_{еф} = (144400 - 49824) * 0,96 = \sim 90793\ грн$$

Економічна ефективність щодо захисту інформації на підприємстві за два роки становить:

$$Е_{еф} = (З_{вт} * 2 - У) * К_{еф} = (144400 * 2 – 94591) * 0,96 = 186440\ грн$$

В результаті підприємство отримає економічний ефект за два роки у вигляді економії фінансів внаслідок запобігання наслідків від крадіжки конфіденційної інформації, фальсифікації інформації, зараження шкідливими програмами, збоїв роботи інформаційної системи та ін. в розмірі 186440 грн. На рисунку 5.2 представлені основні показники економічної ефективності створення мережі Honeynet.

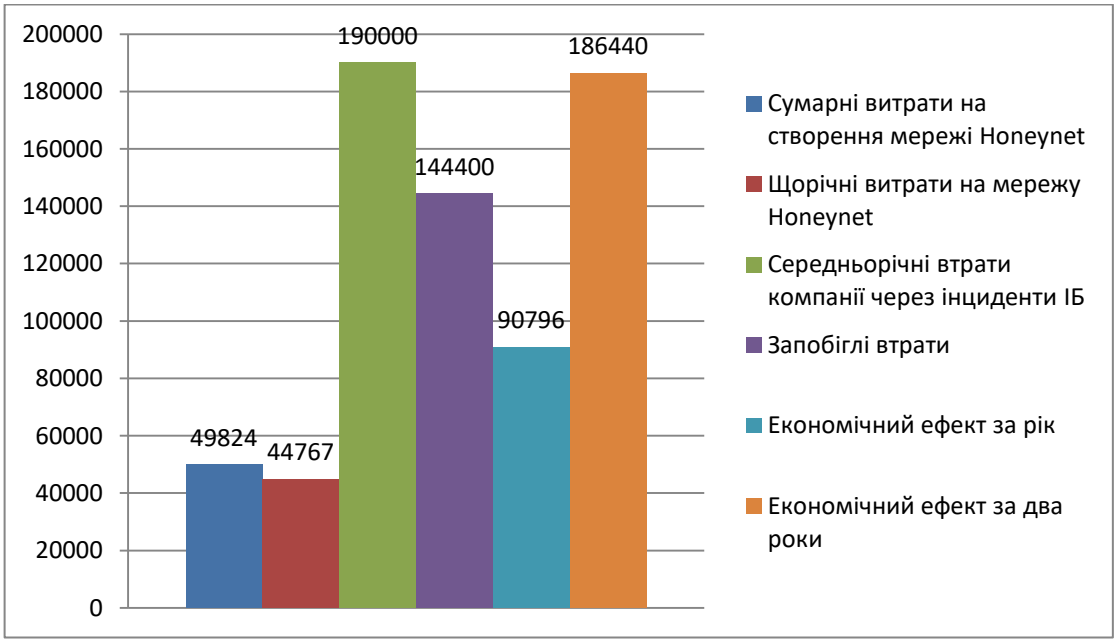

Рисунок 5.2 – Показники економічної ефективності створення мережі Honeynet

Період окупності спроектованої мережі Honeynet для організації не перевищує одного року, тому її впровадження є економічно доцільним.

6 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

6.1 Охорона праці

Дослідження методів розробки адаптивних контейнерних приманок для моніторингу кіберінцидентів проводилось в приміщенні, обладнаному ПЕОМ. Тому надзвичайно важливим фактором безпеки праці є дотримання правил користування технікою, норм та правил охорони праці. Основними потенційними небезпеками при проведенні робіт з ПЕОМ в приміщенні (офісі), а також заходами для їх усунення є:

1) Негативний вплив напруженості та інтенсивності трудових факторів на психофізіологічний стан працівника, що може призвести до зниження працездатності або психологічних перевантажень.

Для мінімізації негативного впливу напруженості та інтенсивності праці в роботі передбачені:

- оптимальний режим трудового процесу;
- впровадження сучасних методів психологічного розвантаження.

Для підвищення ефективності праці застосовують заходи щодо оптимізації менеджменту організації праці, які передбачають системи тренінгів, навчання керівного складу організації, підвищення організаційної культури тощо.

2) Можливість враження електричним струмом внаслідок порушень правил електробезпеки або несправності електроприладів, що може призвести до електричних травм або летального випадку.

Для виключення можливості ураження електричним струмом передбачені:

- організаційні заходи – вивчення та атестація на знання правил електробезпеки;
- технічні заходи – устрій системи заземлення електроспоживаючого обладнання згідно «Правилам улаштування електроустановок» («ПУЕ»)

[101]; забезпечення недоступності струмопровідних частин для випадкового доторкання; використання ізоляції; використання методів колективного захисту від ураження електричним струмом: захисного заземлення, занулення та автоматичного відключення; періодична перевірка опору заземлення; контроль та профілактика пошкоджень ізоляції.

3) Недостатнє освітлення робочої зони внаслідок несправності освітлювальних приладів, що може призвести до погіршення зору.

Для забезпечення оптимальної освітленості в робочих приміщеннях яка нормується згідно з ДБН В.2.5-28:2018 «Природне і штучне освітлення» [102], як 200 лк передбачено устрій бокового природного освітлення та системи штучного рівномірного загального освітлення. У системі використовуються люмінесцентні лампи типу ЛБ, ЛД потужністю від 40 до 80 Вт, які встановлюються у світильники типу ПВЛ.

4) Незадовільні параметри мікроклімату в приміщенні внаслідок неефективної роботи апаратури для опалення та повітрообміну, що може призвести до загальних захворювань.

Для забезпечення оптимальних параметрів повітряного середовища передбачено виконання вимог ДСН 3.3.6-042-99 «Санітарні норми мікроклімату виробничих приміщень» [103] та ГОСТ 12.1.005-88 (1991) «ССПБ. Загальні санітарно-гігієнічні вимоги до повітря робочої зони» [104].

Для забезпечення оптимальних умов передбачено устрій системи водяного або парового опалення згідно ДБН В.2.5-67:2013 «Опалення, вентиляція та кондиціонування» [105] та встановлення побутового кондиціонера моделі Huyndai ARN07HQBUA.

5) Можливість загоряння внаслідок порушень правил пожежної безпеки, що може призвести до пожеж.

Комплекс протипожежних заходів для приміщення (офісу) обладнаного персональними комп’ютерами з ВДТ розроблений згідно з вимогами НАПБ А.01.001-2014 «Правила пожежної безпеки в Україні» [106].

Виходячи з аналізу речовин та матеріалів, які використовуються при роботі у приміщенні, відповідно до вимог НАПБ Б.03.002-2007 «Визначення категорій приміщень, будинків, установок за вибухопожежною та пожежною небезпекою» [107], приміщення (офіс) обладнане ВДТ належить до виробництв категорії «В» з пожежної небезпеки – простір в приміщенні, в якому знаходяться тверді горючі речовини та матеріали.

Оскільки приміщення (офіс), обладнане ВДТ, належить до виробництв категорії «В» з пожежної небезпеки, тому згідно вимог ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги» [108] воно має II ступінь вогнестійкості.

З технічних та організаційних заходів запобігання пожеж в приміщенні (офісі), обладнаному персональними комп'ютерами з ВДТ, передбачені наступні протипожежні заходи:

а) На силовому обладнанні, силових та освітлювальних колах, згідно з вимогами пункту 3.1 «ПУЕ», встановлені захисні пристрої, що вимикають джерело живлення від ділянки електричного кола, у якій виникло коротке замикання.

б) Згідно з вимогами НАПБ А.01.003-2009 «Правила улаштування та експлуатації систем оповіщення про пожежу та управління евакуацією людей в будинках та спорудах» [109] і ДБН В.2.5-56:2014 «Системи протипожежного захисту» [110], в приміщенні (офісі) обладнаному персональними комп'ютерами з ВДТ встановлена система пожежної й охоронної сигналізації «Сигнал-ВК6». Яка забезпечує виявлення теплових і димових ознак пожежі та місця виникнення пожежі з точністю до місця розміщення датчика.

в) Відповідно до наказу «Про затвердження правил експлуатації та типових норм належності вогнегасників» 15.01.2018 №25 [111], для гасіння електрообладнання у приміщенні (офісі) обладнаному персональними комп'ютерами з ВДТ, що знаходиться під напругою, передбачені вуглекислотні вогнегасники типу ВВК-5 в кількості 2 штук. Відстань між вогнегасниками та місцями можливих загорянь не перевищує 10 м.

Передбачені заходи по забезпеченню безпеки, виробничої санітарії, гігієни праці та пожежної безпеки для приміщення, в якому проводиться дослідження методів розробки адаптивних контейнерних приманок для моніторингу кіберінцидентів, забезпечують безпечні та комфортні умови праці.

6.2 Вплив виробничого середовища на працездатність та здоров’я користувачів комп’ютерів

Вагомий вплив на працездатність та здоров’я користувачів комп’ютерів здійснює виробниче середовище. Це середовище у виробничих приміщеннях (офісах), в основному, визначається мікрокліматом, освітленням, наявністю шкідливих речовин у повітрі, рівнем шуму, випромінювання.

Під виробничим мікрокліматом розуміють стан повітряного середовища виробничого приміщення, який визначається температурою, відносною вологістю, рухом повітря та тепловим випромінюванням нагрітих поверхонь, що в сукупності впливають на тепловий стан організму людини. В процесі трудової діяльності людина перебуває у постійній тепловій взаємодії з виробничим середовищем.

Кількість тепла, що утворюється в організмі, залежить від фізичного навантаження працівника, а рівень тепловіддачі – від мікрокліматичних умов виробничого приміщення. Оскільки робота за комп'ютером характеризується малими фізичними навантаженнями, то цей вид діяльності належить до категорії легких робіт за критерієм енерговитрат організму.

Порушення теплового балансу може призвести до перегрівання або ж переохолодження організму людини і, з рештою, до захворювання.

Віддача тепла організмом людини здійснюється, в основному, за рахунок випромінювання і випаровування вологи з поверхні шкіри. Чим нижча температура повітря і швидкість його руху, тим більше тепла віддається випромінюванням. При високій температурі значна частина тепла втрачається випаровуванням поту.

Вологість повітря істотно впливає на віддачу тепла випаровуванням. Через високу вологість випаровування погіршується і віддача тепла зменшується. Зниження вологості покращує процес тепловіддачі випаровуванням. Однак, надто низька вологість викликає висихання слизових оболонок, їх пересихання та розтріскування, забруднений хвороботворними мікробами.

Рухомість повітря визначає рівень тепловіддачі з поверхні шкіри конвекцією і випаровуванням. Різкі коливання температури в приміщенні, яке продувається холодним повітрям (перетягом) значно порушуючи терморегуляцію організму і можуть викликати простудні захворювання.

Таким чином, для нормального теплового самопочуття людини важливо забезпечити певне співвідношення температури, відносної вологості та швидкості руху повітря, тобто певні мікрокліматичні умови. Такі умови визначаються, в основному, категорією роботи, що виконується, та періодом року і можуть бути оптимальними та допустимими.

Відповідно до ДСанПіН 3.3.2-007-98 [112] у виробничих приміщеннях та робочих місцях з ВДТ та ПК мають забезпечуватись оптимальні значення параметрів мікроклімату.

До категорії Іа належать роботи, що виконуються сидячи і не потребують фізичного напруження, при яких витрати енергії складають до 139 Вт, а до категорії 1б – роботи, що виконуються сидячи, стоячи або пов'язані з ходінням та супроводжуються деяким фізичним напруженням, при яких витрати енергії становлять від 140 до 174 Вт.

Для забезпечення оптимальних мікрокліматичних умов в будь-який період року приміщення, в яких розташовані комп’ютеризовані робочі місця повинні бути обладнані системами опалення. Однак найкраще розв’язання цього питання – це встановлення кондиціонерів, які автоматично підтримують задані параметри мікроклімату.

В повітрі зовнішнього природного середовища, як і в повітряному середовищі приміщень завжди є наявною певна кількість заряджених частинок,

що називаються іонами. Так в 1 $см^3$ чистого зовнішнього повітря міститься близько 1000 негативних іонів і понад 1200 позитивних. Іонний склад повітря може значно змінюватись під впливом цілої низки факторів, до яких також належить специфіка виробничої діяльності. Так, проведені дослідження підтвердили факт суттєвої трансформації іонного складу повітря на робочих місцях з ВДТ протягом виробничої зміни. Встановлено, що вже через 5 хвилин роботи ВДТ концентрація легких негативних іонів знизилася приблизно у 8 разів, а через 3 години роботи – була вже на рівні, близькому до нуля. Істотно знизилась концентрація середніх та важких негативно заряджених частинок. Разом з тим концентрація позитивних іонів зростала і через 3 години роботи з ВДТ у повітрі робочої зони переважали позитивно заряджені частинки усіх розмірів. Така зміна балансу іонного складу повітря призводить до несприятливого впливу на здоров'я користувачів ВДТ. Дослідження, проведені як за кордоном, так і в Україні підтвердили негативний вплив, зумовлений збільшенням кількості позитивних іонів на розумову та фізичну працездатність, розвиток втоми, діяльність серцево-судинної системи бронхо-легеневого апарату, кровотворення, вегетативної нервової системи. Відзначено значний вплив на систему реєстрації інформації, передусім на її найбільш лабільну ланку – короткотермінову пам'ять Водночас, результати проведених досліджень засвідчують сприятливий вплив негативних іонів, що знаходяться в повітрі, на здоров'я людини.

Необхідно зазначити, що проведені дослідження стосовно впливу іонного складу повітря на здоров'я людини, підтвердили положення, висунуті ще на початку XX століття нашим співвітчизником, основоположником геліобіології О.Л. Чижевським.

ДНАОП 0.03-3.06-80 «Санітарно-гігієнічні норми допустимих рівнів іонізації повітря виробничих та громадських приміщень» [113] регламентує рівні іонізації повітря приміщень при роботі за ВДТ та ПК.

Відомо, що шум несприятливо діє на слуховий аналізатор та інші органи та системи організму людини. Визначальне значення щодо такої дії має

інтенсивність шуму, його частотний склад, тривалість щоденного впливу, індивідуальні особливості людини, а також специфіка виробничої діяльності.

Рівні звукового тиску в октавних смугах частот, рівні звуку та еквівалентні рівні звуку на робочих місцях, обладнаних ВДТ і ПК визначені дСанПіН 3.3.2-007-98. Основними заходами та засобами боротьби з шумом є:

- зниження рівнів шуму в джерелі його утворення (застосовується, як правило, в процесі проектування);
- використання звукопоглинальних та звукоізоляційних засобів;
- раціональне планування виробничих приміщень та робочих місць.

На комп'ютеризованих робочих місцях основними джерелами шуму є вентилятори системного блоку, накопичувачі, принтери ударної дії.

Під час виконання робіт ПК у виробничих приміщеннях значення характеристик вібрації на робочих місцях не повинні перевищувати допустимих значень, визначених СН 3044-84 [114].

Для зниження вібрації обладнання, пристрої, пристосування необхідно встановлювати на спеціальні амортизуючі прокладки, передбачені нормативними документами.

Дисплеї на основі ЕПТ є потенційним джерелом випромінювання кількох діапазонів електромагнітного спектра: рентгенівського, оптичного, радіочастотного. Кожний вид випромінювання відрізняється своїми особливими характеристиками впливу на організм людини.

В багатьох країнах світу були проведені дослідження щодо можливого рентгенівського випромінювання відеотерміналів комп'ютерів. Встановлено, що джерелом «м'якого» рентгенівського випромінювання є екран.

Найвищі рівні рентгенівського випромінювання зареєстровані при максимальній яскравості та при щільно заповненому екрані. Однак, у всіх випадках виявлене рентгенівське випромінювання від ВДТ не перевищувало фонового рівня.

Необхідно зазначити, що відповідно до Норм радіаційної безпеки України (НРБУ-97) [115] гранично допустима потужність експозиційної дози та

рентгенівського випромінювання на відстані 5 см від екрана відеотерміналу при будь-яких положеннях регулювальних пристроїв становить 7,74*10-12 А/кг, що відповідає еквівалентній дозі 0,1 мбер/год (100 мкР/год).

Оптичні види випромінювання виникають завдяки взаємодії електронів з шаром люмінофору, нанесеного на екран і ВДТ. Область оптичного випромінювання включає ультрафіолетове (УФ), світлове та інфрачервоне (ІЧ) випромінювання.

З метою профілактики несприятливого впливу електромагнітного випромінювання від ВДТ на користувача необхідно:

- встановити на робочому місці відеотермінал, що відповідає сучасним вимогам стосовно захисту від випромінювань (MPR-II або TCO-95);
- встановити на ВДТ старої конструкції заземлений приекранний фільтр (незаземлений захисний екран відіграє лише декоративну роль щодо захисту від електромагнітного випромінювання);
- не переобтяжувати приміщення значною кількістю робочих місць з ВДТ;
- не концентрувати на робочому місці великої кількості радіоелектронних пристроїв;

Отже, нормальна життєдіяльність людини вагомо залежить від умов зовнішнього середовища, зокрема виробничого. Адже в процесі трудової діяльності на організм людини чиниться своєрідний «тиск» несприятливими виробничими факторами, що прямо чи опосередковано впливають на її здоров’я та працездатність. Серед виробничих факторів прийнято розрізняти небезпечні та шкідливі. Небезпечний виробничий фактор – виробничий фактор, дія якого за певних умов може призвести до травм або іншого раптового погіршення здоров’я працівника. Шкідливий виробничий фактор – виробничий фактор, вплив якого може призвести до погіршення стану здоров’я, зниження працездатності працівника. При зменшенні або усуненні шкідливих та небезпечних виробничих факторів, збільшується ефективність виробництва та покращуються умови праці працівників.

7 ЕКОЛОГІЯ

7.1 Екологізація виробництв

Екологізація виробництв – це розширене відтворення природних ресурсів шляхом удосконалення технології, організації матеріального виробництва, підвищення ефективності праці в екологічній сфері. Під екологізацією виробництва розуміється максимально можливе уподібнення виробничих процесів в цілому і ресурсних циклів зокрема природному кругообігу речовин в біосфері, або це будь-які заходи, що знижують небезпеку виробництва для природи та людини.

Досягнення екологічно збалансованого сталого розвитку забезпечується шляхом проведення еколого-економічної політики, що включає: екологізацію суспільного виробництва; формування збалансованої виробничої структури; включення екологічного імперативу в інвестиційну політику, формування ринку екотехнологій і екопослуг; екологізацію інституційних утворень; розвиток економічних інструментів регулювання природокористування.

Можна виділити наступні основні напрямки екологізації суспільного виробництва:

- збереження і відновлення екологічних систем;
- впровадження прогресивних технологій видобутку природної сировини;
- раціональне використання матеріальних ресурсів;
- створення і впровадження маловідходних і безвідходних виробництв;
- екологічно прийнятне розміщення і територіальна організація виробництва;
- скорочення і ліквідація забруднення навколишнього природного середовища.

Першим ефективним економічним стимулом раціонального природокористування є введення плати за споживання природних ресурсів і за шкідливі впливи на їх стан в процесі виробничо-господарської діяльності.

Другий напрямок екологізації здійснюється при відтворенні основних його фондів. Тут головним господарським важелем стає екологічне опрацювання проектів відтворення основних фондів.

Таким чином, еколого-економічні заходи необхідні для раціоналізації використання природно-сировинних ресурсів і виступають як засіб вирішення протиріччя між зростаючими потребами суспільства в природних ресурсах і обмеженими можливостями природи по їх відтворенню і запасам.

Тут можна виділити кілька аспектів:

- підвищення ступеня видобування корисних копалин з надр землі;
- комплексне перероблення, утилізація всіх компонентів сировини, що видобувається;
- скорочення втрат ресурсів при їх доведенні до споживача;
- вдосконалення структури споживання ресурсів, економія ресурсів, утилізація відходів, що утворюються у населення;
- застосування нових видів енергії та матеріалів.

Принцип безвідходного виробництва запозичений у природних екосистем, які працюють за замкнутою схемою. Кругообіг речовин в природі відтворює життя у всіх її різноманітних формах при повній утилізації відходів. Можна виділити кілька основних принципів екологізації виробництв:

- Розробка та впровадження технологічних процесів і схем, які виключають або доводять до мінімуму відходи і викиди в навколишнє середовище шкідливих речовин, створення водооборотних циклів і безстічних систем для економії та охорони від забруднення шкідливими речовинами прісної води як одного з найбільш дефіцитних ресурсів.
- Проектування і впровадження систем перероблення відходів виробництва і споживання, повернення в основний виробничий цикл вторинних матеріальних ресурсів.
- Розробка та впровадження принципово нових процесів отримання традиційних видів продукції та усунення відходів.

- Створення регіональних промислових комплексів, підприємства яких пов'язані переробкою відходів.

Підтримка життєздатності програми екологізації підприємства вимагає постійної уваги та участі з боку керівництва, а також постійної участі будь-якої людини, залученої в планування та реалізацію даної програми. Перешкоди можуть виявлятися у формі проблем з якістю продукції, одержуваної новими методами, опору частини керівництва та працівників або конфліктом з тими чи іншими положеннями регіонального та федерального законодавства. Проте, за належної підтримки та ентузіазму авторитетних в компанії людей, типографія може реалізувати будь-які проекти. Екологізація виробництва може стати частиною проекту по впровадженню системи управління якістю на підприємстві, забезпечуючи найважливіший показник – якість навколишнього середовища.

7.2 Статистичний аналіз тенденцій і закономірностей динаміки в екології

При аналізі рядів динаміки важливо виявити загальну тенденцію розвитку (тренд) екологічного явища, тобто встановити, в якому напрямку (зростає, зменшується) і за якою залежністю (лінійна чи нелінійна) вона змінюється. Ця задача в статистиці називається вирівнюванням динамічних рядів. Часто рівні ряду з часом змінюються (коливаються), але ця зміна для різних явищ неоднакова і може викликатись різними причинами. Говорять, що динаміка ряду включає три компоненти: тенденцію (або тривало часовий рух); коротко часовий систематичний рух; несистематичний випадковий рух. Статистичне вивчення тенденції ґрунтується на розкладанні динамічного ряду на дві складові:

$$y_t = f(t) = + E_t, \qquad (7.1)$$

де $f(t)$ - основна тенденція, зумовлена впливом постійно діючих чинників; E_t - залишкова величина, що визначає вплив випадкових коливань.

Тенденція *f(t)* виявляється при заміні фактичних рівнів динамічного ряду іншими, обчисленими за певною методикою. Останні порівняно з первинними мають значно меншу варіацію, завдяки чому тенденція стає наочною.

Вивчаючи ряди динаміки, дослідники намагаються виявити головним чином загальну тенденцію (тренд) у змінах рівнів ряду, тобто основну закономірність розвитку явища, яка вільна від дії різних випадкових факторів.

Загальною тенденцією динаміки є послідовне прагнення до росту, стабільності або зниженню рівнів. Тенденцію визначають за характером змін показників динаміки – абсолютного приросту, темпу приросту і т.д. На основі сформованої тенденції оцінюється загальна 192 закономірність розвитку. Для виявлення тенденції ряди динаміки підлягають спеціальній обробці – вирівнюванню. Вона дозволяє характеризувати особливості зміни за часом динамічного ряду в найбільш загальному вигляді, вважаючи, що через фактор часу можна передати вплив усіх головних факторів.

До способів і методів вирівнювання динамічних рядів можуть бути віднесені такі:

а) укрупнення інтервалів;

б) згладжування способом ковзної (плинної, рухомої) середньої;

в) аналітичне вирівнювання.

Укрупнення інтервалів є найпростішим способом вирівнювання рядів. Сутність способу укрупнення інтервалів полягає у заміні вихідного ряду динаміки іншим, показники якого зображають явище за більш тривалий період часу: річні інтервали замінюються за двох, трьох або п’ятирічний.

Внаслідок укрупнення інтервалів відхилення, які викликані дією випадкових факторів, взаємно гасяться, згладжуються і більш ясно виявляються в дії основні фактори зміни рівнів, тобто загальна тенденція зменшення викидів.

Згладжування способом ковзної середньої є одним з ефективних методів виявлення загальної тенденції розвитку явища в часі. Суть його полягає в тому, що первинні рівні динамічного ряду замінюються середніми по інтервалах.

Спочатку середній рівень обчислюється з певного числа перших рівнів ряду, потім – з такої самої кількості рівнів, але починаючи з другого, далі – починаючи з третього і так далі. Розраховані таким чином середні рівні ряду ніби ковзають по ряду динаміки від його початку до кінця, при цьому щоразу відкидається один рівень спочатку і додається наступний. Звідси наша — «ковзна» (рухома) середня. Згладжування таким способом можна здійснювати за будь-яким числом членів ряду.

Аналітичне вирівнювання рядів динаміки вважається найбільш удосконаленим способом оброблення ряду з метою встановлення кількісного вираження тенденції розвитку явища. Завдання такого вирівнювання полягає у знаходженні простої математичної формули (апроксимуючої функції), яка найкраще зображала б загальну тенденцію ряду динаміки. Рівні ряду тут розглядаються як функція часу, а завдання (вирівнювання) зводиться до визначення виду функції, її параметрів за емпіричними даними та обчислення теоретичних рівнів за знайденою формулою.

Аналіз коливань і сталості динамічних рядів. У розвитку екологічних процесів поєднуються необхідність і випадковість, тому поряд з тенденцією їм притаманні відхилення від тренда, сезонні коливання, структурні зрушення тощо. Для вимірювання коливань рівнів динамічного ряду використовують абсолютні та відносні похибки й характеристики варіації:

- абсолютна похибка: $\varepsilon = /y_t - Y_t/$ (7.2)
- відносна похибка: $\varepsilon_y = /y_t - Y_t/: y_t$ (7.3)
- амплітуда (розмах) коливань: $R_t = \varepsilon_{max} - \varepsilon_{min}$ (7.4)
- середнє квадратичне відхилення: $\sigma = \sqrt{(yt - Yt)^2}$ (7.5)
- коефіцієнт варіації: $V_t = \sigma / \bar{y_t} \times 100$ (7.6)

Протилежна мінливості властивість – сталість. Мірою сталості служить різниця $100 - V_t$. Чим ближчий цей коефіцієнт до 100%, тим вища сталість динамічного ряду.

Оцінка адекватності (надійності) обраного рівняння аналітичного вирівнювання здійснюється за допомогою середньої відносної помилки апроксимації:

$$E = ½ \sum(|y_t - Y_t|) / y_t \times 100 \qquad (7.7)$$

Цей коефіцієнт розраховується для всіх рівнянь. Порівняння отриманих коефіцієнтів дає підстави вважати адекватним те рівняння, у якого середня помилка апроксимації є меншою. Отже, коефіцієнт апроксимації дає змогу оцінити правильність установлення характеру тенденції і надійність вибраного рівняння вирівнювання.

Методи кореляційно-регресійного аналізу, за допомогою яких проводять аналітичне вирівнювання рядів динаміки, слугують важливим інструментом передбачення значень майбутніх рівнів.

Однак прогнози, складені на основі моделей аналітичного вирівнювання, можна вважати надійними, якщо виконуються, принаймні, такі вимоги:

- обсяг вибірки, який використаний для побудови рівняння регресії, є достатньо великим;
- значення рівнів ряду динаміки змінюється повільно;
- відсутній вплив випадкових факторів.

При прогнозуванні використовують різні способи:

- спосіб найменших квадратів використовується, якщо стабільна абсолютна швидкість зміни рівнів;
- середній абсолютний приріст – якщо стабільні абсолютні прирости;
- середній коефіцієнт зміни – якщо стабільні коефіцієнти зміни.

Якщо на досліджуваному інтервалі часу спостерігається тенденція зміни коефіцієнтів рівняння тренду, то застосування регресійних моделей для обчислення прогнозних значень рядів динаміки може призвести до хибних результатів. Ефективнішим методом прогнозування рядів динаміки в таких випадках є метод експоненційного згладжування Брауна.

ВИСНОВКИ

Основним внеском в цій дипломній роботі стало розроблення нового контейнерного монітора кіберінцидентів, який керується мережею приманок, який дозволяє системним адміністраторам представляти дані загроз осмисленим чином, забезпечуючи цілісне уявлення про свої системи.

Існують переконливі докази того, що монітор інцидентів, розроблений в цьому дослідженні, забезпечує більшу придатність даних приманок, ніж той, який був би отриманий тільки при їх використанні:

- Візуалізація даних приманок означає, що немає потреби переглядати великі обсяги журналів, щоб зрозуміти найбільш актуальну інформацію про загрози;
- агрегування даних з декількох приманок за допомогою централізованої системи реєстрації даних, що дозволяє легко ідентифікувати тенденції загроз;
- Можливість отримувати миттєві оповіщення про загрози по електронній пошті;
- Встановлення та налаштування приманок та мережі, необхідної для їх підключення, повністю автоматизоване.

Таким чином, ця розробка ефективно усуває багато з основних перешкод на шляху до використання активного мережевого захисту в інфраструктурі, що, безумовно, є цінним внеском в область кібербезпеки.

Цінність рішення мережі приманок, яка була реалізована в цій роботі, значно підвищило використання контейнерів. Вони служать перевіркою концепції щодо зручності використання активних механізмів захисту мережі для мережевих адміністраторів, надаючи можливість автоматичного встановлення і налаштування кількох середовищ «на льоту».

Як видно з цього дослідження, контейнеризація додатків безпеки викликає досить багато супутніх проблем, основним чином через відсутність інструкцій та ресурсів, якими можна було б керуватись при розробці таких

систем. Тому ця область сповнена напрямків для досліджень і оцінки різних підходів.

Робота з приманками може бути вкрай непередбачуваною, що ускладнює проведення переконливих досліджень в умовах браку часу. Однак, очевидно, що здатність забезпечувати активний захист мережі за допомогою обману і моніторингу дій зловмисників робить їх привабливим варіантом для боротьби з динамічними загрозами, які все виникають все частіше. Об’єкти критичної інфраструктури гостро потребують адаптивних можливостей приманок до виявлення загроз для захисту своїх систем і тих, хто від них залежить.

Навіть з обмежених експериментів, які проводилися в цьому дослідженні, ясно, що ботнети IoT є надзвичайно активними: кожна «річ» з підключенням до Інтернету є засобом використання комунікацій та взаємодії. Люди все більше залежать від взаємопов'язаних служб і додатків і, як наслідок, стають вразливими для кіберзагроз, які перебувають поза досяжністю їх власних пристроїв.

Об’єкти критичної інфраструктури будуть піддаватися серйозним кібератакам і надалі, оскільки кібервійни між різними державами у світі тривають. Однак, перш ніж ці проблеми зможуть бути вирішені, є необхідними величезні інновації в галузі безпеки IoT: тільки коли система спроектована з урахуванням вимог безпеки, її можна вважати відносно безпечною. Замість цього додатки продовжують отримувати інтегровану в них можливість з'єднання з дуже невеликою увагою до безпеки. Цілком ймовірно, що до тих пір, поки у виробників не з'явиться дійсно ринковий стимул для впровадження гідних механізмів безпеки у свої продукти, ситуація навколо пристроїв «Інтернету речей» навряд чи зміниться.

БІБЛІОГРАФІЯ

1. National Cyber Security Centre UK, Russian Military 'Almost Certainly' Responsible for Destructive 2017 Cyber Attack - NCSC Site. URL: https://www.ncsc.gov.uk/news/russian-military-almost-certainly-responsibledestructive-2017-cyber-attack (дата звернення: 18.10.2019).
2. United States Department of Justice, Memorandum for Heads of Department Components: Cyber-Digital Task Force. URL: https://www.justice.gov/file/1035457/download (дата звернення: 18.10.2019).
3. Krebs B. U.K. Hospitals Hit in Widespread Ransomware Attack. URL: https://krebsonsecurity.com/2017/05/u-k-hospitals-hit-in-widespread-ransomwareattack/ (дата звернення: 18.10.2019).
4. Oracle Corp. Anatomy of a Cyber Attack: The Lifecycle of a Security Breach. URL: http://www.oracle.com/us/technologies/linux/anatomy-of-cyber-attacks-wp-4124673.pdf (дата звернення: 19.10.2019).
5. Khalimonenko A., Kupreev O., Ilganaev K. DDoS attacks in Q4 2017. Securelist. URL: https://securelist.com/ddos-attacks-in-q4-2017/83729/ (дата звернення: 19.10.2019).
6. Bastos M. T., Mercea D. The Brexit Botnet and User-Generated Hyperpartisan News. Social Science Computer Review. URL: https://journals.sagepub.com/doi/10.1177/0894439317734157 (дата звернення 19.10.2019).
7. Synopsys Inc. Heartbleed Bug. URL: http://heartbleed.com/ (дата звернення: 19.10.2019).
8. Sicari S., Rizzardi A., Grieco L. A., Coen-Porisini A. Security, Privacy and Trust in Internet of Things: The Road Ahead. Vol. 76. Computer Networks. 2015. pp. 146– 164.
9. Statista GmbH. IoT: Number of Connected Devices Worldwide 2012-2025. Statista. URL: https://www.statista.com/statistics/471264/iot-number-of-connecteddevices-worldwide/ (дата звернення: 20.10.2019).

10. United Nations, 2017b. World Population Prospects: The 2017 Revision, Key Findings and Advance Tables. ESA/P/WP/248. United Nations. URL : https://esa.un.org/unpd/wpp/publications/Files/WPP2017_KeyFindings.pfg (дата звернення: 20.10.2019).
11. Lackorzynski T., Koepsell S. Hello Barbie - Hacker Toys in a World of Linked Devices in Broadband Coverage in Germany // ITG-Symposium, 2017. C.1-7.
12. Insecam.org, Insecam - World biggest online cameras directory. URL: https://www.insecam.org/, 2017 (дата звернення: 20.10.2019).
13. Farrell S. LPWAN Overview Internet-Draft draft-ietf-lpwan-overview-10. URL: https://tools.ietf.org/html/draft-ietf-lpwan-overview-10 (дата звернення: 21.10.2019).
14. Dowling S., Schukat M., Melvin H. A ZigBee Honeypot to Assess IoT Cyberattack Behaviour // 28th Irish Signals and Systems Conf. Ireland, 2017. pp.1-6.
15. Telnet Protocol Specification. RFC 854. 05.1983. 14 с.
16. Shodan, default password - Shodan. URL: https://www.shodan.io/search?query=default+password (дата звернення: 22.10.2019)
17. Schneier B. The Internet of Things Is Wildly Insecure - And Often Unpatchable. URL: https://www.wired.com/2014/01/theres-no-good-way-to-patchthe-internet-of-things-and-thats-a-huge-problem/ (дата звернення: 22.10.2019).
18. Lonvick C. M., Ylonen T. The Secure Shell (SSH) Authentication Protocol. RFC 4252, 01. 2006.
19. Krebs B. Krebs on Security - In-depth Security News and Investigation. URL: https://krebsonsecurity.com/ (дата звернення: 22.10.2019).
20. Antonakakis M., April T., Bailey M., Bernhard M., Bursztein E., Cochran J., Durumeric Z., J., Halderman A., Invernizzi L., Kallitsis M., Kumar D., Lever C., Ma, J. Mason Z., Menscher D., Seaman C., Sullivan N., Thomas K., Zhou Y. Understanding the Mirai Botnet. In 26th USENIX Security Symposium (Vancouver, BC) Vancouver. 2017. pp. 1093–1110.

21. Schneier B. Botnet of Things. URL: https://www.schneier.com/essays/archives/2017/03/botnets of things.html, 2017 (дата звернення: 23.10.2019).
22. Krebs B. Who is Anna Senpai, the Mirai Worm Author?. URL: https://krebsonsecurity.com/2017/01/who-is-anna-senpai-the-mirai-worm-author/ (дата звернення: 23.10.2019).
23. Goodin D. BrickerBot, the permanent denial-of-service botnet, is back with a vengeance — Ars Technica. URL: https://arstechnica.com/informationtechnology/2017/04/brickerbot-the-permanent-denial-of-service-botnet-is-backwith-a-vengeance/ (дата звернення: 23.10.2019).
24. The Janit0r, Internet Chemotherapy. URL: https://archive.is/PQAnU#selection13.5156-13.5200 (дата звернення: 23.10.2019).
25. United States Computer Emergency Readiness Team, HIDDEN COBRA - North Korea's DDoS Botnet Infrastructure. URL: https://www.uscert.gov/ncas/alerts/TA17- 164A (дата звернення: 25.10.2019).
26. Beyza Unal P. L. Cybersecurity of Nuclear Weapons Systems: Threats, Vulnerabilities and Consequences, Tech. Rep. MSU-CSE-06-2, International Security Depart ment, The Royal Institute of International Affairs, Chatham House, 10 St James's Square, London SW1Y 4LE, 06.2019.
27. Brenner J. Keeping America Safe: Towards more Secure Networks for Critical Sectors. Report on a Series of MIT Workshops, 2015-2016. URL: https://internetpolicy.mit.edu/reports/Report-IPRI-CIS-CriticalInfrastructure-2017-Brenner.pdf (дата звернення: 25.10.2019).
28. Scaife N., Traynor P., Butler K., Making Sense of the Ransomware Mess (and Planning a Sensible Path Forward). Vol. 36. IEEE Potentials. 2017. C. 28–31.
29. Sir Amyas Morse, Investigation: WannaCry Cyber Attack and the NHS, 2017. URL: https://www.nao.org.uk/wp-content/uploads/2017/10/Investigation-WannaCry-cyber-attack-and-the-NHS.pdf (дата звернення: 25.10.2019).

30. Smart W. Lessons learned: Review of the WannaCry Ransomware Cyber Attack. Independent report. URL: https://www.england.nhs.uk/wp-content/uploads/2018/02/lessons-learned-review-wannacry-ransomware-cyber-attack-cio-review.pdf (дата звернення: 25.10.2019).
31. Voronkov A., Iwaya L. H., Martucci L. A., Lindskog S. Systematic Literature Review on Usability of Firewall Configuration, ACM Comput. Surv., vol. 50, pp. 87:1–87:35, 12.2017.
32. O. Santos O. End-to-End Network Security: Defense-in-Depth. Cisco Press. URL: http://index-of.co.uk/Hacking-Coleccion/End-to-End%20Network%20Security%20-%20Defense-in-Depth.pdf (дата звернення: 26.10.2019).
33. Bellovin S. Thinking Security: Stopping Next Year's Hackers. Addison-Wesley Professional, 2015. 400 с.
34. Schneider F. Blueprint for a Science of Cybersecurity. Technnical report. URL: https://www.cs.cornell.edu/fbs/publications/SoS.blueprint.pdf (дата звернення: 27.10.2019).
35. Wang C., Lu Z. Cyber Deception: Overview and the Road Ahead. IEEE Security & Privacy, vol. 16, с. 80–85, 03.2018.
36. Mokube I., Adams M. Honeypots: Concepts, Approaches, and Challenges. Proceedings of the 45th Annual Southeast Regional Conference (New York, NY, USA) New York, 2007. С.321-326.
37. Nawrocki M., Wahlisch M., Schmidt T. C., Keil C., Schonfelder J. A Survey on Honeypot Software and Data Analysis, CoRR, vol. abs/1608.06249, 2016.
38. Spitzner L. Honeypots: Catching the Insider Threat. Proceedings of the 19th Annual Computer Security Applications Conference (Washington, DC, USA) Washington, 2003. 170 с.
39. Canary - Know when it Matters. Thinkst Applied Research: веб-сайт. URL: https://canary.tools/, 2018. (дата звернення: 28.10.2019).
40. McFarland B. Ethical Deception and Preemptive Deterrence in Network Security. Technical report. URL: https://www.sans.org/reading-room/whitepapers/firewalls/paper/1616 (дата звернення: 28.10.2019)

41. Spitzner L. Honeypots: Are They Illegal? — Symantec Connect. Technical article. URL: https://www.symantec.com/connect/articles/honeypots-are-they-illegal (дата звернення: 02.11.2019)
42. Sasse M. A., Smith M. The Security-Usability Tradeoff Myth [Guest editors' introduction]. IEEE Security & Privacy. Vol. 14. 2016. С. 11–13.
43. Vasilomanolakis E., Karuppayah S., Kikiras P., Muhlhauser M. A Honeypotdriven Cyber Incident Monitor: Lessons Learned and Steps Ahead. In Proceedings of the 8th International Conference on Security of Information and Networks (New York, NY, USA) New York, 2015. С. 158–164.
44. Biswas S. An Introduction to the Docker Ecosystem - Boolean World. Technical article. URL: https://www.booleanworld.com/introduction-docker-ecosystem/ (дата звернення: 03.11.2019)
45. Leung A., Spyker A., Bozarth T. Titus: Introducing Containers to the Netflix Cloud. Commun. ACM. Vol. 61. 2018. С. 38–45.
46. Combe T., Martin A., Pietro R. D. To Docker or Not to Docker: A Security Perspective. Vol. 3. IEEE Cloud Computing. 2016. С. 54–62.
47. Tamminen U. desaster/kippo: Kippo - SSH Honeypot. GitHub: веб-сайт. URL: https://github.com/desaster/kippo (дата звернення: 04.11.2019)
48. GitHub - cuckoosandbox/cuckoo: Cuckoo Sandbox is an Automated Dynamic Malware Analysis System. GitHub: веб-сайт. URL: https://github.com/cuckoosandbox/cuckoo (дата звернення: 05.11.2019)
49. Barron T., Nikiforakis N. Picky attackers: Quantifying the role of system properties on intruder behavior. In Proceedings of the 33rd Annual Computer Security Applications Conference (New York, NY, USA) New York, 2017. С. 387–398.
50. T-Pot 17.10 - Multi-Honeypot Platform rEvolution. DTAG Community Honeypot Project: вебсайт. URL: http://dtag-dev-sec.github.io/mediator/feature/2017/11/07/tpot-17.10.html (дата звернення: 05.11.2019)

51. Trost J. Modern Honey Network — Anomali. Technical article. URL: https://www.anomali.com/blog/mhn-modern-honey-network (дата звернення: 06.11.2019)
52. GitHub - cowrie/cowrie: Cowrie SSH Honeypot (based on kippo) with SCP, SFTP, exec command, direct-tcpip support and many other features. Github: веб-сайт. URL: https://github.com/cowrie/cowrie (дата звернення: 07.11.2019).
53. Oosterhof M. Cowrie Honeypot - Security Intelligence. Technical article. URL: http://www.micheloosterhof.com/cowrie/ (дата звернення: 07.11.2019).
54. Edwards S., Profetis I. Hajime: Analysis of a decentralized internet worm for IoT devices. Technical report. URL: http://security.rapiditynetworks.com/publications/2016-10-16/hajime.pdf (дата звернення: 07.11.2019).
55. GitHub - cowrie/docker-cowrie: Cowrie Docker GitHub repository. GitHub: вебсайт. URL: https://github.com/cowrie/docker-cowrie (дата звернення: 07.11.2019).
56. Provos N. Honeyd General Information. Technical article. URL: http://www.honeyd.org/general.php (дата звернення: 07.11.2019)
1. 57 Provos N. Honeyd: A Virtual Honeypot Daemon. Technical article. URL: https://www.usenix.org/conference/12th-usenix-security-symposium/honeyd-virtual-honeypot-daemon (дата звернення: 08.11.2019).
57. Pa Y., Suzuki S., Yoshioka K., Matsumoto T., Kasama T., Rossow C. IoTPOT: A Novel Honeypot for Revealing Current IoT Threats. Vol. 24. Journal of Information Processing. 2016. C. 522–533.
58. Leita C., Dacier M., Wicherski G. SGNET: A Distributed Infrastructure to handle Zero-Day Exploits. Techical report. URL: http://www.eurecom.fr/en/publication/2164/download/ce-leitco-070201.pdf (дата звернення: 08.11.2019).
59. Luo T., Xu Z., Jin X., Jia Y., Ouyang X. IoTCandyJar: Towards an IntelligentInteraction Honeypot for IoT Devices. Technical article.URL: https://www.blackhat.com/docs/us-17/thursday/us-17-Luo-Iotcandyjar-

Towards-An-Intelligent-Interaction-Honeypot-For-IoT-Devices.pdf (дата звернення: 08.11.2019).

60. Pisarcik P., Sokol P. Framework for Distributed Virtual Honeynets. In Proceedings of the 7th International Conference on Security of Information and Networks, SIN ’14, (New York, NY, USA), New York, 2014. C.324:324–324:329.
61. Kedrowitsch A., Yao D. D., Wang G., Cameron K. A First Look: Using Linux Containers for Deceptive Honeypots. In Proceedings of the 2017 Workshop on Automated Decision Making for Active Cyber Defense, SafeConfig@CCS 2017, (Dallas, TX, USA. October 30 - November 03, 2017), Dallas, 2017. C. 15–22.
62. Miramirkhani N., Appini M. P., Nikiforakis N., Polychronakis M. Spotless Sandboxes: Evading Malware Analysis Systems Using Wear-and-Tear Artifacts. In 2017 IEEE Symposium on Security and Privacy, 2017. C. 1009–1024.
63. DTAG Community Honeypot Project, T-Pot 16.10 - Multi-Honeypot Platform Redefined. URL: http://dtag-dev-sec.github.io/mediator/feature/2016/10/31/t-pot16.10.html (дата звернення 10.11.2019)
64. Mansfield-Devine S. The Right Response: How Organisations should React to Security Incidents Network Security. Vol. 12. 2017. pp. 16 – 19.
65. Cito J., Ferme V., Gall H. C. Using Docker Containers to Improve Reproducibility in Software and Web Engineering Research. In Web Engineering (Bozzon, Alessandro and Cudre-Maroux, Philippe and Pautasso, Cesare, ed.). 2016. pp. 609–612.
66. Amazon Web Services, Inc., What is AWS? - Amazon Web Services. URL: https://aws.amazon.com/what-is-aws/ (дата звернення: 10.11.2019).
67. Mohallel A. A., Bass J. M., Dehghantaha A. Experimenting with Docker: Linux Container and Base OS Attack Surfaces. In 2016 International Conference on Information Society. 2016. pp. 17–21.
68. Desaster, FAQ - desaster/kippo Wiki. Github: вебсайт. URL: https://github.com/desaster/kippo/wiki/FAQ (дата звернення: 10.11.2019).

69. Elastic Co., Secure communication with Logstash by using SSL — Filebeat Reference [6.2] — Elastic. URL: https://www.elastic.co/guide/en/beats/filebeat/current/configuring-ssllogstash.html (дата звернення: 11.11.2019) .
70. Elastic Co. Introduction — X-Pack for the Elastic Stack [6.2] — Elastic. Technical article. URL: https://www.elastic.co/guide/en/x-pack/current/xpack-introduction.html (дата звернення: 11.11.2019)
71. Docker Inc. Docker Security. Docker Documentation. URL: https://docs.docker.com/engine/security/security/ (дата звернення: 12.11.2019).
72. Chelladhurai J., Chelliah P. R., Kumar S. A. Securing Docker Containers from Denial of Service (DoS) Attacks. International Conference on Services Computing (SCC). 2016. pp. 856–859.
73. Docker Inc., Docker Hub. URL: https://hub.docker.com/ (дата звернення: 12.11.2019)
74. Vlasenko D. BusyBox - The Swiss Army Knife of Embedded Linux. URL: https://www.busybox.net/downloads/BusyBox (дата звернення: 12.11.2019).
75. Oosterhof M., cowrie/INSTALL.md at master - cowrie/cowrie. Github: веб-сайт. URL: https://github.com/cowrie/cowrie/blob/master/INSTALL.md (дата звернення: 13.11.2019).
76. Reitz K. Pipenv and Virtual Environments - The Hitchhiker’s Guide to Python. URL: http://docs.python-guide.org/en/latest/dev/virtualenvs/ (дата звернення: 13.11.2019).
77. Cisco Systems Inc. TP-Link TL-SC3171 IP Cameras Remote Command Injection Vulnerability URL: https://tools.cisco.com/security/center/viewAlert.x?alertId=32038 (дата звернення: 14.11.2019).
78. MaxMind Inc. GeoIP2 Database Demo — MaxMind. URL: https://www.maxmind.com/en/geoip-demo (дата звернення: 14.11.2019)

79. Docker Inc. Dockerfile Reference. Technical documentation. URL: https://docs.docker.com/engine/reference/builder/ (дата звернення: 14.11.2019).
80. Ferme V. Using Docker Containers to Improve Reproducibility in Software and Web Engineering. Technical article. URL: https://www.slideshare.net/vincenzoferme/using-dockercontainers-to-improve-reproducibility-in-software-and-web-engineering (дата звернення: 14.11.2019).
81. Docker Inc. Use bridge networks. Technical documenation. URL: https://docs.docker.com/network/bridge/ (дата звернення: 15.11.2019).
82. Docker Inc. Use overlay networks. Technical documentation. URL: https://docs.docker.com/network/overlay/ (дата звернення: 15.11.2019).
83. Amazon Web Services, Inc. Regions and Availability Zones - Amazon Elastic Compute Cloud. User guide. URL: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regionsavailability-zones.html (дата звернення: 16.11.2019).
84. Elasticsearch BV. Heap: Sizing and Swapping — Elasticsearch: The Definitive Guide [2.x] — Elastic. User Guide. URL: https://www.elastic.co/guide/en/elasticsearch/guide/current/heap-sizing.html (дата звернення: 16.11.2019)
85. Anicas M. How To Install Elasticsearch, Logstash, and Kibana (ELK Stack) on Ubuntu 14.04 — DigitalOcean. Technical article. URL: https://www.digitalocean.com/community/tutorials/how-to-install-elasticsearchlogstash-and-kibana-elk-stack-on-ubuntu-14-04 (дата звернення: 16.11.2019).
86. Moskowitz R. G., Karrenberg D., Rekhter Y., Lear E., Groot G. J. Address Allocation for Private Internets. RFC 1918. 1996.
87. Gerhards R. The Syslog Protocol. RFC 5424. 2009.
88. Kernc/logkeys: A GNU/Linux keylogger that worked!. GitHub: веб-сайт. URL: https://github.com/kernc/logkeys (дата звернення: 16.11.2019).

89. dumpkeys(1) - Linux manual page. User guide. URL: http://man7.org/linux/manpages/man1/dumpkeys.1.html (дата звернення: 18.11.2019).
90. Command line - How do I log all input and output in a terminal session? Ask Ubuntu. URL: https://askubuntu.com/questions/161935/how-do-i-log-all-input-andoutput-in-a-terminal-session (дата звернення: 18.11.2019).
91. NGINX Inc., Welcome to NGINX Wiki!. NGINX. URL: https://www.nginx.com/resources/wiki/ (дата звернення: 19.11.2019).
92. Dominiguez F. Logging Cowrie logs to the ELK stack. Blog. URL: http://blog.fernandodominguez.me/logging-cowrie-logs-to-the-elk-stack/ (дата звернення: 20.11.2019).
93. Huawei Technologies Co. Security Notice - Statement on Remote Code Execution Vulnerability in Huawei HG532 Product. Technical review. URL: http://www.huawei.com/en/psirt/security-notices/huawei-sn-20171130-01-hg532- en (дата звернення: 21.11.2019).
94. Check Point Software Technologies Ltd. CPAI-2017-1016 — Check Point Software. Technical article. https://www.checkpoint.com/defense/advisories/public/2017/cpai-2017-1016.html (дата звернення: 21.11.2019).
95. Martz M. Sivel/speedtest-cli: Command line interface for testing internet bandwidth using speedtest.net. Sivel. URL: https://github.com/sivel/speedtest-cli (дата звернення: 21.11.2019).
96. Chin W. Y., Markatos E. P., Antonatos S., Ioannidis S. HoneyLab: LargeScale Honeypot Deployment and Resource Sharing. Third International Conference on Network and System Security, 2018. Oct 2018. pp. 381–388.
97. Docker Inc. Limit a container’s resources. Docker Documentation. URL: https://docs.docker.com/config/containers/resource constraints/ (дата звернення: 23.11.2019).
98. Amazon Web Services, Inc. AWS IP Address Ranges. Amazon Web Services. URL: https://docs.aws.amazon.com/general/latest/gr/aws-ip-ranges.html (дата звернення: 23.11.2019).

99. Про авторське право і суміжні права: Закон України від 23.12.1993 № 3792-12 (у редакції від 11.07.2001 № 2627-14).
100. Правила улаштування електроустановок. Затв. пост. Каб. Мін. України 29.04.2017. Київ, 2017. 617 с.
101. ДБН В.2.5-28:2018. Природне і штучне освітлення. Київ, 2018. 133 с.
102. ДСН 3.3.6-042-99. Санітарні норми мікроклімату виробничих приміщень. Київ, 1999. 10 с.
103. ГОСТ 12.1.005-88. ССПБ. Загальні санітарно-гігієнічні вимоги до повітря робочої зони. Москва, 1988. 95 с.
104. ДБН В.2.5-67:2013. Опалення, вентиляція та кондиціонування. Київ, 2013. 141 с.
105. НАПБ А.01.001-2014. Правила пожежної безпеки в Україні. Київ, 2014. 85 с.
106. ДСТУ Б В.1.1-36:2016. Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою. Київ, 2016. 31 с.
107. ДБН В.1.1-7:2016. Пожежна безпека об’єктів будівництва. Загальні вимоги. Київ, 2016. 36 с.
108. НАПБ А.01.003-2009. Правила улаштування та експлуатації систем оповіщення про пожежу та управління евакуацією людей в будинках та спорудах. Київ, 2016. 16 с.
109. ДБН В.2.5-56:2014. Системи протипожежного захисту. Київ, 2014. 127 с.
110. Про затвердження правил експлуатації та типових норм належності вогнегасників: Закон України від 15.01.2018 №25. Київ, Парламентське видавництво, 2018. 41 с.
111. ДСанПіН 3.3.2-007-98. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин. Київ, 1998. 48 с.
112. ДНАОП 0.03-3.06-80. Санітарно-гігієнічні норми допустимих рівнів іонізації повітря виробничих та громадських приміщень. СРСР, 1980. 65 с.

113. СН 3044-84. Санітарні норми вібрації робочих місць. СРСР, 1984. 33с.
114. НРБУ-97. Норм радіаційної безпеки України. Київ, 1997. 127 с.