КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

УДК 004.4 https://doi.org/10.32515/2664-262X.2025.11(42).1.56-67

М. С. Продеус, **А. О. Нічепорук**, доц., канд. техн. наук, **А. С. Каштальян**, доц., канд. техн. наук
Хмельницький національний університет, м. Хмельницький, Україна
e-mail: mprodeus99@ukr.net, andrey.nicheporuk@gmail.com, yantonina@ukr.net

Система моніторингу, виявлення, реагування та захисту інформації на основі Honeypot-файлів

З огляду на зростання кіберзагроз та обмежену ефективність традиційних засобів безпеки, у статті запропоновано систему моніторингу, виявлення, реагування та захисту інформації на основі Honeypot-файлів. Запропонований підхід поєднує розподілене розміщення honeypot-файлів, багаторівневий моніторинг і автоматизоване блокування загроз, що дозволяє знизити кількість помилкових спрацьовувань до 10% та забезпечити реакцію системи із середнім часом відповіді 2 секунди. Розглянуто особливості інтеграції такої системи у корпоративні середовища, виклики її розгортання та можливості масштабування до 50 honeypot-файлів у локальних і хмарних інфраструктурах. Отримані результати демонструють ефективність підходу в ідентифікації та блокуванні вірусів, троянських програм, програм-вимагачів та інших кіберзагроз, що робить його перспективним для посилення захисту корпоративних мереж.

файли-приманки, honeypot, мережева безпека, виявлення загроз, кіберзахист, корпоративні мережі

Постановка проблеми. Для покращення безпеки та захисту інформації корпоративних мереж поряд зі стандартними засобами доцільно використовувати засоби і системи на основі обманних технологій. Як правило це спеціально створені системи, які імітують реальні сервери або пристрої користувачів, щоб привертати увагу хакерів. Їх називають приманками і вони містять фальшиві дані або сервіси, які виглядають привабливо для зловмисників, але не становлять реальної цінності для організації. Основна функція приманок – фіксувати спроби несанкціонованого доступу та аналізувати дії зловмисників [1].

Використання приманок залишається дискусійним питанням у сфері кібербезпеки. Вони приваблюють фахівців своєю простотою впровадження та здатністю вчасно виявляти вторгнення в мережу. Однак проблема виникає, коли організації розглядають приманки як основний засіб виявлення загроз, а не як частину комплексної стратегії захисту.

Одним із видів кіберприманок є приманний файл. Це спеціально створений документ, призначений для виявлення несанкціонованого доступу та моніторингу поведінки потенційних зловмисників. Розміщення такого файлу в системі дозволяє отримувати сповіщення про спроби доступу, що допомагає виявляти загрози безпеці [2]. Приманний файл працює як цифрова пастка. Для звичайних користувачів він залишається непомітним, але привертає увагу кіберзлочинців. Кожна спроба доступу фіксується, що дає змогу відстежувати підозрілу активність і аналізувати потенційні загрози. Це не лише сприяє ранньому виявленню атак, а й дає цінну інформацію про дії зловмисників, що допомагає вдосконалювати механізми кібербезпеки [3].

Ефективність приманок залежить не лише від їхньої правильної конфігурації, а й від стратегічного розташування в мережі. Важливо мінімізувати кількість хибних спрацювань і зробити систему максимально реалістичною, щоб вона залишалася привабливою для кіберзлочинців. Водночас приманки мають бути лише одним із компонентів загальної стратегії кібербезпеки. Для ефективного протистояння атакам, зокрема програмам-вимагачам, необхідно чітко розуміти можливості та обмеження цього інструмента [4].

Аналіз останніх досліджень і публікацій. На сьогодні проблема створення

кіберприманок для аналізу зловмисної активності привертає значну увагу в науковій літературі. У дослідженні [5] автори представляють HoneyHive – розподілену систему виявлення загроз, що базується на приманках для IoT. Робота описує підхід до інтеграції систем приманок у середовище Інтернету речей (IoT) для ефективного моніторингу та аналізу потенційних атак. Основна ідея полягає в розгортанні невеликих пристроїв з вбудованими модулями приманок у різних сегментах мережі, що імітують вразливі вузли, які можуть стати цілями атак.

Однією з ключових переваг HoneyHive є її розподілена архітектура, що дозволяє здійснювати моніторинг безпеки мережі в режимі реального часу без необхідності втручання адміністратора. Поєднання легковагових модулів приманок із централізованим аналізом загроз робить цю систему ефективним рішенням для захисту IoT-інфраструктури. Окрему увагу також приділено питанню поширення метаморфних вірусів, проте активне відстеження таких загроз залишається значним викликом.

У роботі [6] автори розробляють концептуальну модель багатокомп'ютерних систем, призначених для підтримки роботи антивірусних приманок та пасток для виявлення зловмисного програмного забезпечення й кібератак у корпоративних мережах. Запропонований підхід спрямований на запобігання та протидію проникненню метаморфних вірусів. Розроблена модель поєднує комбіновані приманки та пастки, а також контролер ухвалення рішень для виявлення та нейтралізації зловмисного програмного забезпечення й кіберзагроз. Крім того, такі системи можуть включати модифіковані операційні середовища, що виконують програми у зміненому режимі з дослідницькою метою. Запропонований метод підвищує ефективність виявлення та запобігання поширенню метаморфних вірусів у корпоративних мережах.

Дослідження [7] пропонує новий підхід до виявлення метаморфних вірусів. Методика складається з двох етапів: на першому етапі здійснюється пошук еквівалентних функціональних блоків на основі статистичної оцінки частоти інструкцій, а на другому – уточнюється вибір еквівалентних блоків і визначаються рівні обфускації.

Результати дослідження свідчать, що запропонований метод ефективно виявляє метаморфні віруси, навіть якщо вони використовують складні техніки обфускації. Це досягається шляхом аналізу відповідностей між функціональними блоками в різних версіях вірусу, що дозволяє ідентифікувати зловмисне програмне забезпечення, незважаючи на його модифікації. У результаті ця методика сприяє підвищенню надійності кібербезпекових систем у боротьбі із сучасними загрозами.

Постановка завдання. Метою дослідження є розробка та оцінка ефективності системи моніторингу, виявлення, реагування та захисту інформації на основі Honeypot-файлів. Особлива увага приділяється тому, наскільки ефективніше вдосконалене рішення мінімізує помилкові спрацьовування та забезпечує оперативне блокування зловмисників у порівнянні з традиційним підходом Canarytokens.

Основні завдання дослідження: 1) проаналізувати традиційний підхід до створення файлів-приманок за допомогою Canarytokens та виокремити його недоліки; 2) запропонувати систему моніторингу, виявлення, реагування та захисту інформації на основі Honeypot-файлів, що заснована на механізмах динамічних файлів honeypot; використані розподіленої файлової системи honeypot, а також автоматизоване реагування на спроби доступу; 3) провести експериментальне дослідження щодо ефективності запропонованої системи.

Виклад основного матеріалу. Розроблений фреймворк призначений для виявлення вторгнень зловмисних програм. Зокрема, підсистема здатна ідентифікувати віруси та трояни, які намагаються отримати доступ до файлів, що містять конфіденційну інформацію. Вірус, який копіює або змінює документи, буде виявлено

за допомогою моніторингу часу зміни файлу. Зловмисне троянське програмне забезпечення, яке сканує систему на наявність критичних файлів, може запустити документ honeypot, що призведе до його ідентифікації. Шпигунські програми, які передають файли на віддалені сервери, можна виявити через повторні спроби отримати доступ до honeypot [9].

Програми-хробаки, які автоматично поширюються мережами, можуть інфікувати файл honeypot, дозволяючи відстежувати їхню діяльність. Часті зміни файлів можуть свідчити про наявність зловмисного коду, що поширюється мережею. Автоматичне блокування інфікованого пристрою через брандмауер Windows може допомогти стримати поширення загрози [10].

Файл honeypot також можна використовувати для виявлення програм-вимагачів (наприклад, WannaCry, Ryuk, LockBit) [11]. Якщо зловмисне програмне забезпечення намагається зашифрувати файл honeypot, система зареєструє зміни в його структурі. Часті спроби модифікації протягом короткого періоду можуть свідчити про атаку програм-вимагачів [12]. Автоматичне блокування підозрілих процесів або мережевої активності може допомогти запобігти подальшому поширенню програм-вимагачів.

Зловмисники, які отримують початковий доступ до системи, часто сканують файлові сервери на наявність цінних даних. Файли Honeypot, розміщені в різних місцях (локальні диски, хмарні служби, спільні папки), можуть допомогти виявити аномальну поведінку користувачів. Якщо невідомий пристрій спробує отримати доступ до файлу, система може повідомити адміністратора та заблокувати IP-адресу. Якщо файл honeypot використовується в корпоративних спільних середовищах (таких як SharePoint або OneDrive), він може допомогти виявити атаки соціальної інженерії, коли зловмисник намагається отримати доступ до внутрішніх документів [13]. Сповіщення про доступ до файлу honeypot з невідомої IP-адреси можуть вказувати на компрометацію облікового запису. Виконання макросів у файлі може виявити зловмисні документи, які намагаються завантажити додаткові зловмисні програми [14].

Особливо складним типом зловмисного програмного забезпечення є метаморфічні віруси. Метаморфічний вірус може спробувати змінити файл honeypot, щоб вставити свій код або активувати зловмисну логіку. Система виявляє такі зміни через тригер модифікації файлу та реєструє вторгнення. Якщо вірус змінює структуру файлу honeypot (наприклад, вводить новий зловмисний код), це буде записано [15]. Якщо метадані файлу (наприклад, час модифікації або права доступу) змінено, система honeypot також виявить це. Метаморфічний вірус може спробувати скопіювати файл honeypot до тимчасових каталогів або змінити його дозволи. У таких випадках активується тригер копіювання файлу або зміни дозволу, що дозволяє виявити незвичну активність у системі. Деякі метаморфічні віруси використовують для поширення локальну або мережеву реплікацію. Якщо файл honeypot розміщено в мережевому середовищі (наприклад, хмарне сховище або сервер), будь-які спроби завантажити або передати його між хостами можуть бути виявлені за допомогою тригера доступу до мережі [16].

Якщо зловмисне програмне забезпечення працює виключно в пам’яті, не змінюючи файли, його буде важко виявити. Крім того, якщо файлову систему обійти, а вірус працює лише через інфіковані процеси, файли honeypot можуть не активуватися. Методи обфускації коду також можуть ускладнити виявлення на основі вмісту файлу.

Стандартний метод створення приманки має певні обмеження, які можуть знизити його ефективність. По-перше, зловмисники можуть ідентифікувати статичні файли honeypot, якщо їм бракує достатньо реалістичних атрибутів, таких як регулярні оновлення вмісту та зміни метаданих [17]. По-друге, традиційні документи honeypot

реагують лише на певні типи доступу і не враховують більш складні атаки, такі як загрози на основі пам'яті або методи обходу файлової системи. Крім того, ці приманки можуть генерувати велику кількість хибних спрацьовувань, особливо коли законні користувачі або автоматизовані процеси випадково взаємодіють з ними. Це може перевантажити системи безпеки та ускладнити аналіз загроз [18].

Таким чином для вдосконалення технології запропоновано наступні дії: 1) динамічні файли honeypot; 2) використання розподіленої файлової системи honeypot; 3) автоматизоване реагування на спроби доступу.

Замість статичних документів слід використовувати динамічні файли, які автоматично змінюють метадані, такі як дата створення, час останнього доступу або дата останньої зміни. Це підвищує довіру до файлу та ускладнює визначення його як приманки. Розміщення файлів honeypot у різних місцях, таких як внутрішні сервери, хмарне сховище та загальнодоступні ресурси, дозволяє відстежувати дії зловмисників у різних сегментах мережі. Після виявлення неавторизованого доступу система може автоматично активувати додаткові заходи безпеки, такі як блокування підозрілих IP-адрес, активація розширеного моніторингу або сповіщення аналітиків кібербезпеки для негайного реагування. Підсистема розділена на ключові сегменти, кожен з яких покращує протоколи безпеки, що ускладнює ідентифікацію файлу honeypot серед інших даних, рисунок 1.

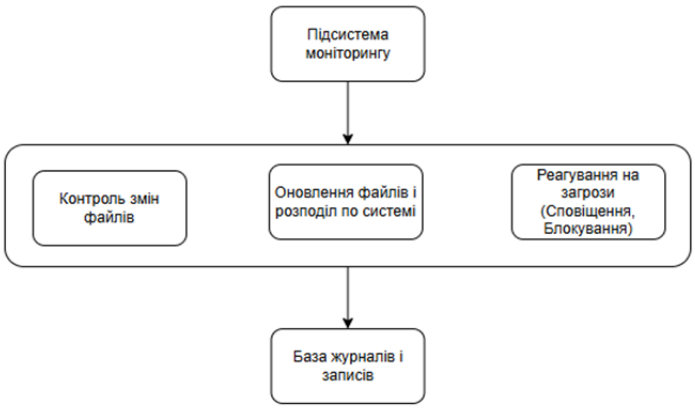


Рисунок 1– Підсистема автоматичного реагування на загрози

Джерело: розроблено авторами

Усі ці дії разом допоможуть підвищити надійність системи та покращити захист від несанкціонованого втручання зловмисників, рисунок 2.

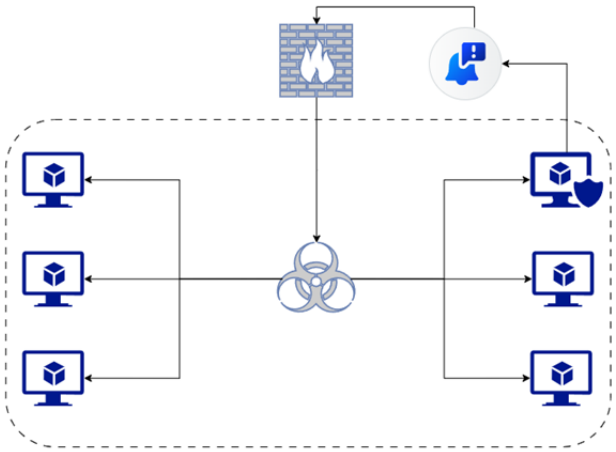


Рисунок 2 – Схема виявлення потенційних загроз за допомогою файлу honeypot

Джерело: розроблено авторами

Динамічні файли honeypot. Динамічні файли-приманки покликані усунути головний недолік традиційних honeypot-документів – їхню статичність. Зловмисники можуть легко виявити такі файли, якщо їхній вміст або мета-дані залишаються незмінними протягом довгого часу. Щоб цього уникнути, необхідно зробити файл таким, що постійно змінюється, не втрачаючи при цьому своєї правдоподібності.

Для реалізації цього підходу можна використовувати Windows із вбудованими інструментами або мову Python. Оптимальним варіантом є створення скрипта, який автоматично оновлюватиме мета-дані файлу, вноситиме незначні зміни у вміст і забезпечуватиме відстеження всіх взаємодій із ним.

Спочатку необхідно підготувати сам honeypot-файл. Його слід зберегти в місці, де до нього можуть отримати доступ потенційні зловмисники, наприклад, у спільній папці або у файловому сховищі організації, рівняння 1. Якщо E(f)=0, то файл створюється зі стандартним вмістом S0: f=S0, де S0 ="Confidential Data - Access Restricted".

$$E(f) = \begin{cases} 1, & \text{якщо файл } f \text{ існує} \\ 0, & \text{якщо файл } f \text{ не існує} \end{cases}, \quad (1)$$

Після цього потрібно реалізувати механізм періодичної зміни мета-даних файлу. У Python це можна зробити за допомогою модуля os, який дозволяє змінювати дату останнього редагування. Наприклад, скрипт може щодня оновлювати атрибут "Last Modified", щоб файл виглядав активним у використанні [19].

Наступним кроком буде внесення випадкових змін у сам вміст файлу. Це можна зробити шляхом додавання невеликих коментарів або змінних фраз у прихованих полях документа. Якщо використовується текстовий файл, можна додати кілька випадкових рядків, які не впливатимуть на загальний вигляд документа, але робитимуть його змінним з точки зору системи. У кінець файлу додається випадковий рядок R, який генерується за формулою 2, де U(a,b) - рівномірний розподіл випадкового числа в інтервалі [1000, 9999].

$$R = "\#LogEntry:" + U(1000{,}9999)\,, \quad (2)$$

де U(a,b) - рівномірний розподіл випадкового числа в інтервалі [1000, 9999]. Після цього новий вміст файлу можна записати як рівняння 3, де Sold – попередній вміст файлу, а Snew – оновлений вміст.

$$S_{new} = S_{old} + R. \quad (3)$$

Після впровадження цих змін слід налаштувати автоматичний запуск скрипта. Це можна представити, як дату останнього редагування файлу T(f) оновлюється до поточного часу Tnow, рівняння 4, в якому Tnow = timestamp(t), а t – поточний момент часу.

$$T(f) = T_{now}. \quad (4)$$

Таким чином, файл-приманка буде постійно оновлюватися, а будь-який доступ до нього залишатиме цифровий слід, що дасть змогу аналізувати активність потенційних зловмисників.

Використання розподіленої файлової системи honeypot. Розподілена система файлів-приманок покликана покращити ефективність honeypot-документів шляхом їх розміщення у різних частинах ІТ-інфраструктури. Якщо файл-приманка знаходиться лише в одному місці, наприклад, на внутрішньому сервері, зловмисник може уникнути взаємодії з ним, і це зменшить його ефективність. Щоб зробити приманку більш привабливою та підвищити ймовірність її використання зловмисниками, потрібно створити копії файлу та поширити їх у різних середовищах, таких як локальні диски, хмарні сховища, внутрішні файлові сервери або корпоративні платформи для обміну документами [20].

Скрипт створює копії файлу-приманки та розміщує їх у кількох місцях у системі. Перед розповсюдженням файл отримує невеликі зміни, щоб уникнути ідентичності всіх копій, що дозволить зробити їх більш правдоподібними. Після цього

він розповсюджується на заздалегідь визначені локації, такі як папки спільного доступу або каталоги, що синхронізуються з хмарними сховищами.

Щоб уникнути зайвого дублювання файлів, скрипт перевіряє, чи файл уже існує в зазначеному місці, і лише потім здійснює його оновлення або перезапис. Під час кожного запуску він змінює мета-дані всіх копій файлу, додаючи новий випадковий рядок у його вміст, відповідно до рівнянь 1-4, після чого автоматично розміщує оновлені версії на заданих шляхах, рівняння 5. Файл-приманка розповсюджується у множину місць P = {p1,p2,…pn}, де кожен файл fi у рі отримує власний унікальний рядок Ri.

$$S(f_i) = S_0 + R_i, \ \forall_i \in [1, \mathrm{n}]. \quad (5)$$

Автоматизоване реагування на спроби доступу. Автоматизоване реагування на спроби доступу до файлу-приманки дозволяє своєчасно виявляти потенційні загрози та запобігати подальшій зловмисній діяльності в мережі. Якщо файл honeypot відкривається або копіюється, система повинна негайно зафіксувати цю подію, надіслати відповідне сповіщення адміністраторам безпеки та вжити додаткових заходів, таких як блокування IP-адреси атакуючого або активація розширеного моніторингу активності користувача [21].

Фреймворк працює у фоновому режимі та контролює файл-приманку, відстежуючи його відкриття, зміну або копіювання. Якщо хтось взаємодіє з файлом, програма записує цю подію в журнал безпеки та надсилає сповіщення на вказану електронну адресу або у системний лог. Фреймворк постійно перевіряє час останньої модифікації файлу-приманки f. Час модифікації визначається як T(f), а поточний час - Tnow рівняння 6.

$$\Delta T = T_{now} - T(f). \quad (6)$$

Якщо різниця ΔT змінюється, це означає, що файл був відкритий або змінений, рівняння 7.

$$\Delta T > 0 \Rightarrow \text{файл змінено}. \quad (7)$$

Якщо виявлено зміну файлу, вона записується у журнал подій L, який містить список часових міток модифікацій L = {T1,T2,…,Tn}. При кожному новому доступі додається запис L = L∪{Tnow}, де Tnow - момент останньої зміни файлу.

Крім того, фреймворк може автоматично додати IP-адресу атакуючого до списку заблокованих у брандмауері Windows, що запобігатиме подальшим спробам доступу до мережі [22]. IP-адреса зловмисника визначається через системний журнал. Позначимо її як IPattacker. Якщо файл змінено більше ніж N разів за проміжок часу ΔTblock, IP-адреса атакуючого додається у список заблокованих адрес B, рівняння 8.

$$\text{Якщо } |L| > N \text{ за } \Delta T_{block}, \text{ то } B = B \cup \{\mathrm{IP}_{attacker}\}. \quad (8)$$

Після цього фреймворк додає правило у брандмауер Windows Fblock(IPattacker) = 1, де Fblock - функція блокування доступу через Windows Firewall.

Об’єднуючи всі вищеописані дії, функцію моніторингу можна представити як рівняння 9.

$$\begin{cases} \text{Якщо } \Delta T > 0, \text{ то } L = L \cup \{T_{now}\} = 1 \\ \text{Якщо } |L| > N \text{ за } \Delta T_{block}, \text{ то } F_{block}\{\mathrm{IP}_{attacker}\} = 1 \end{cases}. \quad (9)$$

Експериментальне дослідження ефективності. Першим етапом у процесі впровадження honeypot-файлів є створення спеціального файлу, що містить псевдоінформацію, яка може зацікавити зловмисників. Після його створення йому присвоюється атрибут lure, що вказує на його приманкову природу. Однак статичний honeypot-файл може швидко втратити свою ефективність, оскільки зловмисники можуть ідентифікувати його як підробку. Тому для підвищення правдоподібності файлу до нього підключаються спеціальні сценарії, які забезпечують його динамічність та періодичні зміни вмісту. Це робить приманку більш реалістичною та знижує ймовірність її ідентифікації як пастки. У випадку з операційною системою Windows

цей процес можна реалізувати за допомогою планувальника завдань, створивши відповідне завдання, яке періодично виконуватиме певний сценарій, що оновлює honeypot-файл через задані інтервали часу [23].

Наступним важливим кроком є поширення honeypot-файлів у системі. Для того щоб автоматизувати цей процес, було використано мову програмування Python, яка дозволяє гнучко керувати створенням та розповсюдженням таких файлів. З метою запобігання надмірному навантаженню системи, було встановлено обмеження на максимальну кількість honeypot-файлів, що можуть одночасно існувати в системі – їхня кількість не перевищує 50 примірників. Це рішення дозволяє мінімізувати зайве споживання ресурсів та підтримувати ефективність системи. Для забезпечення безперервного оновлення honeypot-файлів у Windows було використано планувальник завдань, який періодично виконує певний сценарій, що автоматично додає або оновлює honeypot-файли в системі. Наприклад, завдання може запускатися кожну годину або щодня, що гарантує їхню постійну ротацію та збільшує ймовірність взаємодії зловмисників із ними [24].

Окрім створення та поширення honeypot-файлів, необхідно впровадити механізм контролю доступу до них. Для цього використовуються політики сповіщень, які можуть бути вбудованими в операційну систему або ж реалізованими через сторонні засоби моніторингу. Такі механізми дозволяють миттєво реагувати на спроби несанкціонованого доступу, а також вести журнал активності для подальшого аналізу загроз.

Останнім етапом налаштування є автоматизація відповіді на виявлені тригери. У разі спрацювання honeypot-файлу активуються відповідні сценарії, які можуть виконувати різні дії – від сповіщення адміністратора до автоматичного збору інформації про зловмисника (рисунок 3). Це дозволяє не лише виявляти спроби злому, а й аналізувати поведінку атакувальників, що допомагає покращити безпеку системи в майбутньому.

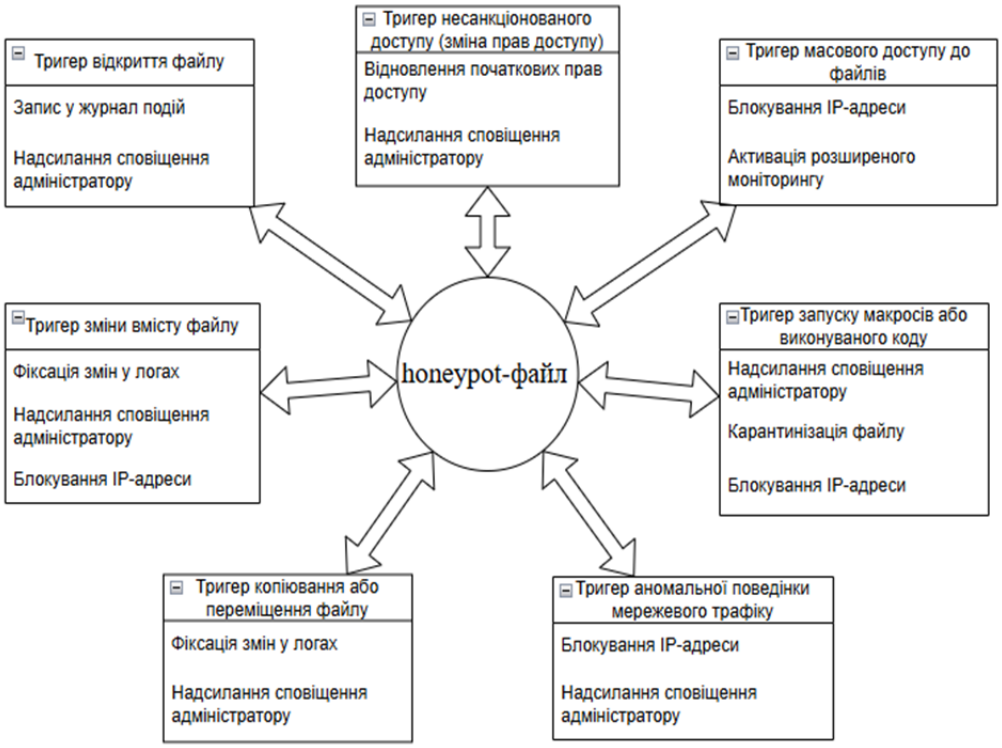
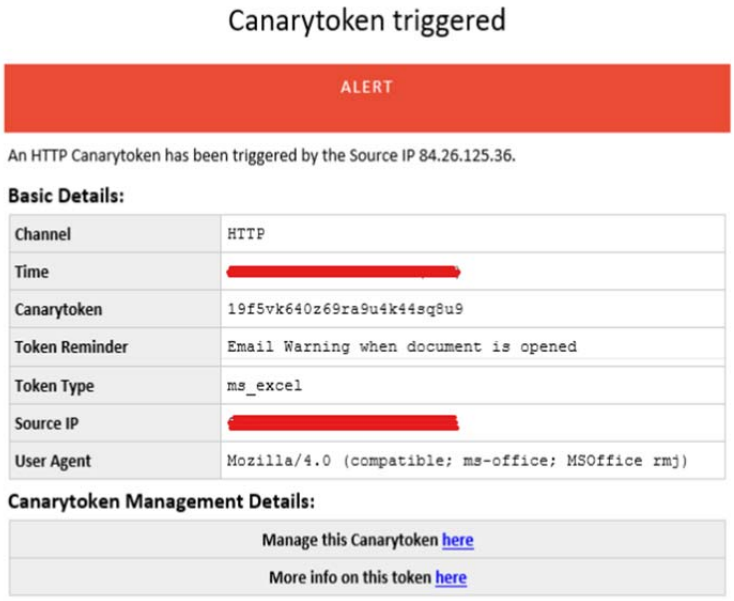


Рисунок 3 – Тригери файлів Honeypot і реакції на них

Джерело: розроблено авторами

Щоб механізм відповіді працював ефективно, необхідно запускати скрипт у фоновому режимі під час запуску операційної системи. Наприклад, це можна зробити за допомогою планувальника завдань Windows, який забезпечить автоматичний запуск програми після перезавантаження комп'ютера. Завдяки такому підходу система зможе миттєво реагувати на підозрілі дії, не вимагаючи постійного втручання адміністратора [25]. Таким чином, якщо хтось спробує завантажити позначений файл, сповіщення буде надіслано на пов'язану електронну адресу, і щоб перевірити систему на вторгнення, буде достатньо переглянути сповіщення, як показано на рис. 4.


Canarytoken triggered

ALERT

An HTTP Canarytoken has been triggered by the Source IP 84.26.125.36.

Basic Details:

Channel	HTTP
Time	[redacted]
Canarytoken	19f5vk640z69ra9u4k44sq8u9
Token Reminder	Email Warning when document is opened
Token Type	ms_excel
Source IP	[redacted]
User Agent	Mozilla/4.0 (compatible; ms-office; MSOffice rmj)

Canarytoken Management Details:

Manage this Canarytoken here

More info on this token here

Рисунок 4 – Електронна пошта зі сповіщенням про завантаження файлу

Джерело: розроблено на основі ресурсу Canarytoken

Щоб оцінити ефективність підсистеми, 50 файлів honeypot було розгорнуто в різних середовищах, включаючи локальні сервери, хмарне сховище та мережеві ресурси. За 30 днів було зафіксовано 125 спроб доступу, з яких 80 були визначені як потенційно зловмисні. З таблиці 1 видно, що запропоноване рішення мало хибнопозитивний рівень 10%, тоді як стандартний файл Canarytokens без удосконалень показав у середньому 30%, що свідчить про високий рівень точності виявлення загроз. Система автоматичного блокування IP для зловмисників спрацьовувала в 95% випадків, забезпечуючи оперативне реагування на загрози. Моніторинг активності показав, що середній час реакції системи на спробу доступу до файлу становить 2 секунди, що дозволяє вчасно вжити заходів безпеки. Як видно з таблиці 2, щонайменше чотири з семи тригерів можуть бути активовані для кожного типу загрози, що допомагає зменшити кількість помилкових спрацьовувань і забезпечує відповіді лише на потенційні загрози.

Таблиця 1 – Порівняння методів

Метод	Хибні спрацювання	Ефективність	Час спрацювання	Надійність
Стандартний підхід	30%	60%	1с	70%
Запропонований фреймворк	10%	95%	2с	95%

Джерело: розроблено авторами

Таким чином, можна відзначити, представлена система забезпечила стабільну роботу без перевантаження ресурсів, а її інтеграція з механізмами кібербезпеки значно знизила ризики компрометації даних.

Таблиця 2 – Відповідність тригерів різним типам загроз

Тригери / Загрози	Віруси	Троянський кінь	Шпигунські програми	Черви	Програми-вимагачі	Метаморфні віруси
Тригер відкриття файлу	Так	Так	Так	Ні	Так	Ні
Тригер зміни вмісту файлу	Так	Так	Ні	Ні	Так	Так
Тригер копіювання або переміщення файлу	Так	Ні	Так	Так	Так	Ні
Тригер несанкціонованого доступу (зміна прав доступу)	Так	Так	Ні	Так	Так	Так
Тригер масового доступу до файлів	Ні	Ні	Так	Так	Так	Так
Тригер ненормальної поведінки мережевого трафіку	Ні	Так	Так	Так	Ні	Так
Тригер для запуску макросів або виконуваного коду	Так	Так	Ні	Ні	Так	Так

Джерело: розроблено авторами

Висновки. У роботі представлено архітектуру розподіленої системи для виявлення кіберзагроз на основі динамічних файлів honeypot, яка поєднує автоматизований моніторинг, розподілене розміщення honeypot-файлів і миттєве реагування на підозрілу активність. Запропонований підхід ефективно ідентифікує широкий спектр загроз, зокрема віруси, трояни, програми-вимагачі та метаморфічні віруси, використовуючи динамічно змінювані файли з реалістичними метаданими.

1. Досліджено класичний підхід до створення файлів-приманок за допомогою Canarytokens. Виявлено, що традиційні методи мають певні обмеження, такі як статичність файлів, обмежена інтеграція з системами захисту та висока ймовірність помилкових спрацьовувань. Це знижує ефективність їх використання у складних кіберзагрозах, особливо при атаках із застосуванням автоматизованих або адаптивних алгоритмів.

2. Запропоновано удосконалену систему захисту, яка базується на динамічних honeypot-файлах та розподіленій файловій системі honeypot. Вона комбінує тригери, такі як зміни вмісту файлу, аномальний мережевий трафік та виконання макросів, що дозволяє значно зменшити кількість помилкових спрацьовувань (до 10% порівняно з традиційними методами – 30%). Також система забезпечує автоматизоване реагування

на спроби доступу, блокуючи 95% зловмисних IP-адрес через інтеграцію з брандмауером Windows із середнім часом відповіді 2 секунди.

3. Проведено експериментальне тестування показало, що розподілене розміщення до 50 honeypot-файлів у різних середовищах (локальні сервери, хмарні сховища, корпоративні платформи) значно підвищує ймовірність виявлення загроз. Водночас система має певні обмеження – вона менш ефективна проти шкідливого програмного забезпечення, що працює виключно в пам'яті або обходить файлову систему, а також потребує вдосконалення алгоритмів аналізу для детектування обфусцованого коду.

Запропоновану систему можна інтегрувати в корпоративні механізми безпеки для покращення захисту від атак соціальної інженерії та автоматизованих кібератак. Майбутні дослідження можуть бути зосереджені на розширенні функціональних можливостей для боротьби зі складнішими загрозами, такими як атаки, керовані штучним інтелектом, а також на оптимізації використання ресурсів під час масштабування системи. Загалом, отримані результати демонструють, що поєднання динамічних технологій honeypot та автоматизації суттєво підвищує рівень кіберзахисту сучасних IT-інфраструктур.

Список літератури

1. Campbell R., Padayachee K., Masombuka T. A survey of honeypot research: Trends and opportunities, *10th International Conference for Internet Technology and Secured Transactions (ICITST*), London, UK, 2015, P. 208-212. doi: 10.1109/ICITST.2015.7412090.
2. Fraunholz D., Schotten H. D. An adaptive honeypot configuration, deployment and maintenance strategy. *International Conference on Cyber Situational Awareness, Data Analytics and Assessment (Cyber SA), IEEE*, 2017. P. 1–8.
3. Pauna A., Patriciu V. V. Enhancing cybersecurity with honeypot systems: A case study. *11th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, 2019. P. 1–6.
4. Rentao Gu., Zeyuan Yang., Yuefeng Ji. Machine learning for intelligent optical networks: A comprehensive survey *Journal of Network and Computer Applications*. 2020.
5. Madison Z. D. Honeyhive – A Network Intrusion Detection System Framework Utilizing Distributed Internet of Things Honeypot Sensors. *Theses and Dissertations*. 2020.
6. Kashtalian A., Lysenko S., Savenko O., Nicheporuk A., Sochor T., Avsiyevych V. Multi-computer malware detection systems with metamorphic functionality. *Radioelectronic and Computer Systems*. 2024. Vol. 1. P. 152–175. DOI: 10.32620/reks.2024.1.13.
7. Savenko O., Lysenko S., Nicheporuk A. Metamorphic viruses' detection technique based on the equivalent functional block search. CEUR-WS. 2017. Vol. 1844. P. 555–569.
8. Canarytokens. Canarytokens – Quick, Free, Detection for the Masses. Retrieved from: https://canarytokens.org/generate.
9. Fraunholz D., Schotten H. D. An adaptive honeypot configuration, deployment and maintenance strategy. *International Conference on Cyber Situational Awareness, Data Analytics and Assessment (Cyber SA),* IEEE, 2017. P. 1–8.
10. Peng Z., Xiaojing G., Surya N., Jianying Z. Modeling social worm propagation for advanced persistent threats. Computers & Security. 2021. P. 102321. DOI: 10.1016/j.cose.2021.102321.
11. Kambourakis G., Kolias C. Honeypots for ransomware detection: A case study on WannaCry and LockBit. *Computers & Security*. 2020. Vol. 95. P. 101823.
12. Lysenko S., Atamaniuk O., Bokhonko O., Vorobiyov V. Method for detection of ransomware cyber threats based on honeypot. CEUR-WS. 2023. P. 300–309.
13. Alsaheel A., Nan Y., Yu L. ATLAS: A practical framework for adaptive threat detection in enterprise environments. *IEEE Symposium on Security and Privacy (SP),* 2021. P. 1–18.
14. Eriksson B., Pellegrino G., Sabelfeld A. Black Widow: Blackbox Data-driven Web Scanning. *Symposium on Security and Privacy (SP),* San Francisco, CA, USA, 2021, IEEE pp. 1125-1142, doi: 10.1109/SP40001.2021.00022.
15. Savenko O., Lysenko S., Nicheporuk A., Savenko B. Approach for the unknown metamorphic virus detection. *8th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems,* Bucharest, 2017. P. 71–76.

16. Markowsky G., Savenko O., Lysenko S., Nicheporuk A. The technique for metamorphic viruses' detection based on its obfuscation features analysis. *CEUR-WS*. 2018. Vol. 2104. P. 680–687.
17. Kambourakis G., Genç Z. Dynamic honeypot configuration to mitigate static detection in ransomware attacks. *Computers & Security*. 2020. Vol. 96. P. 101923.
18. Beuran R., Inoue T., Tan Y. Realistic Cybersecurity Training via Scenario Progression Management. *European Symposium on Security and Privacy Workshops (EuroS&PW)*, Stockholm, Sweden, IEEE 2019, P. 67-76. doi: 10.1109/EuroSPW.2019.00014.
19. Sethuraman S., Jadapalli T., Sudhakaran D. Flow based containerized honeypot approach for network traffic analysis: An empirical study. Computer Science Review. 2023. P. 5–10. doi: 10.1016/j.cosrev.2023.100600.
20. Baykara M., Das R. A novel honeypot based security approach for real-time intrusion detection and prevention systems. *Journal of Information Security and Applications.* 2018. P. 103-116. doi: 10.1016/j.jisa.2018.06.004.
21. Fraunholz D., Zimmermann M., Schotten H. D. SOAR-integrated honeypots for automated threat response. *17th International Conference on Availability, Reliability and Security (ARES),* ACM, 2022. P. 1–10.
22. Nguyen T., Jones M. Automated threat response in honeypot-enabled networks using dynamic firewall rules. *IEEE International Conference on Cyber Security and Resilience (CSR),* IEEE, 2021. P. 1–9.
23. Gupta R., Patel A. Automating security maintenance in Windows environments: A task scheduler approach. *International Conference on Computational Science and Computational Intelligence (CSCI),* IEEE, 2020. P. 1–6.
24. Alotaibi B., Elleithy K. Scalable honeypot deployment using Python scripting for enterprise networks. *Journal of Cybersecurity and Privacy.* 2021. Vol. 1, no. 2. P. 234–250.
25. Johnson L., Martinez C. Persistent security automation in Windows: Leveraging task scheduler for background threat response. *IEEE Symposium on Cybersecurity Applications and Technologies (SCAT),* IEEE, 2022. P. 1–7.

References

1. Campbell, R., Padayachee, K., & Masombuka, T. (2015). A survey of honeypot research: Trends and opportunities. *10th International Conference for Internet Technology and Secured Transactions (ICITST)*, 208–212. https://doi.org/10.1109/ICITST.2015.7412090
2. Fraunholz, D., & Schotten, H. D. (2017). An adaptive honeypot configuration, deployment and maintenance strategy. International Conference on Cyber Situational Awareness, *Data Analytics and Assessment (Cyber SA)*, 1–8.
3. Pauna, A., & Patriciu, V. V. (2019). Enhancing cybersecurity with honeypot systems: A case study. 11th International Conference on Electronics, *Computers and Artificial Intelligence (ECAI)*, 1–6.
4. Gu, R., Yang, Z., & Ji, Y. (2020). Machine learning for intelligent optical networks: A comprehensive survey. *Journal of Network and Computer Applications*, arXiv:2003.05290
5. Madison, Z. D. (2022). Honeyhive – A network intrusion detection system framework utilizing distributed Internet of Things honeypot sensors, Thesis, AD1102962.
6. Kashtalian, A., Lysenko, S., Savenko, O., Nicheporuk, A., Sochor, T., & Avsiyevych, V. (2024). Multi-computer malware detection systems with metamorphic functionality. *Radioelectronic and Computer Systems*, 2024(1), 152–175. https://doi.org/10.32620/reks.2024.1.13
7. Savenko, O., Lysenko, S., & Nicheporuk, A. (2017). Metamorphic viruses' detection technique based on the equivalent functional block search. *CEUR-WS*, 1844, 555–569.
8. Canarytokens. (n.d.). Canarytokens – Quick, free, detection for the masses. Retrieved from https://canarytokens.org/generate
9. Peng, Z., Xiaojing, G., Surya, N., & Jianying, Z. (2021). Modeling social worm propagation for advanced persistent threats. *Computers & Security*, 102321. https://doi.org/10.1016/j.cose.2021.102321
10. Kambourakis, G., & Kolias, C. (2020). Honeypots for ransomware detection: A case study on WannaCry and LockBit. *Computers & Security*, 95, 101823.
11. Lysenko, S., Atamaniuk, O., Bokhonko, O., & Vorobiyov, V. (2023). Method for detection of ransomware cyber threats based on honeypot. *CEUR-WS*, 300–309.
12. Alsaheel, A., Nan, Y., & Yu, L. (2021). ATLAS: A practical framework for adaptive threat detection in enterprise environments. *IEEE Symposium on Security and Privacy (SP)*, 1–18.
13. Eriksson, B., Pellegrino, G., & Sabelfeld, A. (2021). Black Widow: Blackbox data-driven web scanning. *Symposium on Security and Privacy (SP)*, 1125–1142. https://doi.org/10.1109/SP40001.2021.00022
14. Savenko, O., Lysenko, S., Nicheporuk, A., & Savenko, B. (2017). Approach for the unknown metamorphic

virus detection. *8th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems*, 71–76.
15. Markowsky, G., Savenko, O., Lysenko, S., & Nicheporuk, A. (2018). The technique for metamorphic viruses' detection based on its obfuscation features analysis. *CEUR-WS*, 2104, 680–687.
16. Kambourakis, G., & Genç, Z. (2020). Dynamic honeypot configuration to mitigate static detection in ransomware attacks. *Computers & Security*, 96, 101923.
17. Beuran, R., Inoue, T., & Tan, Y. (2019). Realistic cybersecurity training via scenario progression management. *European Symposium on Security and Privacy Workshops (EuroS&PW)*, 67–76. https://doi.org/10.1109/EuroSPW.2019.00014
18. Sethuraman, S., Jadapalli, T., & Sudhakaran, D. (2023). Flow-based containerized honeypot approach for network traffic analysis: An empirical study. *Computer Science Review*, 5–10. https://doi.org/10.1016/j.cosrev.2023.100600
19. Baykara, M., & Das, R. (2018). A novel honeypot-based security approach for real-time intrusion detection and prevention systems. *Journal of Information Security and Applications*, 103–116. https://doi.org/10.1016/j.jisa.2018.06.004
20. Fraunholz, D., Zimmermann, M., & Schotten, H. D. (2022). SOAR-integrated honeypots for automated threat response. 17th International Conference on Availability*, Reliability and Security (ARES)*, ACM, 1–10.
21. Nguyen, T., & Jones, M. (2021). Automated threat response in honeypot-enabled networks using dynamic firewall rules. *IEEE International Conference on Cyber Security and Resilience (CSR)*, 1–9.
22. Gupta, R., & Patel, A. (2020). Automating security maintenance in Windows environments: A task scheduler approach. *International Conference on Computational Science and Computational Intelligence (CSCI)* 1–6.
23. Alotaibi, B., & Elleithy, K. (2021). Scalable honeypot deployment using Python scripting for enterprise networks. *Journal of Cybersecurity and Privacy*, 1(2), 234–250.
24. Johnson, L., & Martinez, C. (2022). Persistent security automation in Windows: Leveraging task scheduler for background threat response. *IEEE Symposium on Cybersecurity Applications and Technologies (SCAT)*, 1–7.

Maksym Prodeus, **Andrii Nicheporuk**, Assoc. Prof., PhD tech. sci., **Antonina Kashtalian,** Assoc. Prof., PhD tech. sci.
Khmelnytskyi National University, Khmelnytskyi, Ukraine

Honeypot-Based Information Monitoring, Detection, Response and Protection System

The increasing complexity of cyber threats poses significant challenges to existing security measures, which often fail to provide sufficient protection. This paper presents an approach to improving cybersecurity through honeypot-based techniques. The study focuses on the development and deployment of decoy files designed to detect unauthorized access attempts and monitor malicious activities. Honeypots play a crucial role in identifying various attack types, including insider threats and masquerade attacks, which traditional security systems often overlook. The paper also discusses the integration of honeypots into comprehensive security frameworks, examining their optimal use cases and effectiveness in real-world applications.

The research explores the design and implementation of an advanced honeypot framework that enhances threat detection and response. The proposed system utilizes dynamic decoy files, which change metadata and content to maintain authenticity and attract malicious actors. Various triggers, such as file access, modification, and unauthorized copying, are employed to detect suspicious behavior. The study also evaluates the effectiveness of automated response mechanisms, including IP blocking and real-time monitoring. The framework's performance is analyzed through experimental deployment across different IT environments, highlighting its advantages over traditional static honeypots. Key performance indicators, including detection accuracy, response time, and false positive rates, are assessed to validate the system's reliability.

The results demonstrate that the proposed honeypot-based system significantly improves threat detection and response capabilities while minimizing false alarms. The integration of dynamic honeypots into corporate cybersecurity infrastructures enhances resilience against cyberattacks, including ransomware and advanced persistent threats. However, certain limitations, such as the inability to detect memory-only malware and highly obfuscated threats, remain challenges for future research. The study concludes that adaptive honeypot strategies, combined with automated threat intelligence, can substantially enhance modern cybersecurity defense mechanisms.

decoy files, honeypot, network security, threat detection, cyber defense, corporate networks

Одержано (Received) 06.03.2025 *Прорецензовано (Reviewed) 11.03.2025*
Прийнято до друку (Approved) 14.03.2025