Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Аналіз методик виявлення вторгнень у системах інформаційної безпеки

Виконав: студент VI курсу, групи СБм-61
спеціальності 125 Кібербезпека
(шифр і назва спеціальності)

_____ Микитишин А.А.
(підпис) (прізвище та ініціали)

Керівник _____ Лечаченко Т.А.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Лечаченко Т.А.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Загородна Н.В.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль - 2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
«__» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека
(шифр і назва спеціальності)

Студенту Микитишину Артуру Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Аналіз методик виявлення вторгнень у системах інформаційної безпеки

Керівник роботи доктор філософії, старший викладач кафедри КБ Лечаченко Тарас Анатолійович
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «16» 11 2023 року № _

2. Термін подання студентом завершеної роботи 25.12.2023р.

3. Вихідні дані до роботи наукові літературні джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
1. Аналіз предметної області. 1.1. Аналітичний огляд. 1.2. Програмні засоби захисту інформаційних систем. 1.3. Система виявлення вторгнень 2. Теоретична частина. 2.1. Аналіз системних журналів. 2.2. Конфігурація відправлення системних журналів. 2.3. Проектування власної системи виявлення вторгнень. 2.4. Виявлення загроз на основі спостерігачів. 2.5. Системи сповіщення для реагування на інциденти. 3. Практична частина. 3.1. Розробка Logstash pipeline для аналізу системних журналів. 3.2. Розробка модулів спостерігачів для виявлення аномалій. 3.3. Розробка розширення RustHound для посилення аналізу. 3.4. Розробка системи сповіщень про інциденти. 4. Безпека життєдіяльності, основи охорони праці. 4.1. Охорона праці. 4.2. Комп'ютерне забезпечення процесу оцінки радіаційної та хімічної обстановки. 5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів). 5.1 Титулка. 5.2 Актуальність. 5.3 Мета дослідження. 5.4 Програмні засоби захисту інформаційних систем. 5.5 Структура IDS. 5.6 Структура SIEM. 5.7 Збір Системних журналів. 5.8 Типи системних журналів. 5.9 Аналіз системних журналів

Збагачення системних журналів за допомогою Rusthound. 5.10 Спостерігачі. 5.11 Система сповіщень. 5.12 Сповіщення про інциденти.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основи охорони праці	Осухівська Г.М., зав. каф. КС		
Безпека життєдіяльності	Стручок В.С., ст. викладач каф. ОХ		

7. Дата видачі завдання 16.11.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	16.11 – 16.11	*Виконано*
2.	Підбір джерел про методи виявлення вторгнень у	17.11 – 20.11	*Виконано*
	системах інформаційної безпеки		
3.	Опрацювання джерел про методи виявлення вторгнень у	21.11 – 23.11	*Виконано*
	системах інформаційної безпеки		
4.	Виконання дослідження щодо методів виявлення	24.11 – 27.11	*Виконано*
	вторгнень у системах інформаційної безпеки		
5.	Підготовка матеріалу	28.11 – 30.11	*Виконано*
6.	Оформлення розділу «Аналіз предметної області»	01.12 – 03.12	*Виконано*
7.	Оформлення розділу «Теоретична частина»	04.12 – 06.12	*Виконано*
8.	Оформлення розділу «Практична частина»	07.12 – 09.12	*Виконано*
9.	Виконання завдання до підрозділу «Безпека	10.11 – 11.12	*Виконано*
	життєдіяльності, основи хорони праці»		
10.	Оформлення кваліфікаційної роботи	12.12 – 13.12	*Виконано*
11.	Нормоконтроль		
12.	Перевірка на плагіат		
13.	Попередній захист кваліфікаційної роботи		
14.	Захист кваліфікаційної роботи	26.12	

Студент _____ *Микитишин А.А.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Лечаченко Т.А.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Аналіз методик виявлення вторгнень у системах інформаційної безпеки // Микитишин Артур Андрійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем та програмної інженерії, кафедра кібербезпеки, група СБм–61 // Тернопіль, 2023 // С. 83, рис. –15, табл. – , слайдів –18.

Ключові слова: ЛОГ-АНАЛІЗ, IDS, СИСТЕМА ЗАХИСТУ, FILEBEAT, RUSTHOUND, JINJA, ELASTICSEARCH, PYTHON, ІНЦИДЕНТ-ВИЯВЛЕННЯ

Кваліфікаційна робота презентує комплексний підхід до виявлення порушень безпеки в інформаційних системах, використовуючи методи лог-аналізу та інтеграції різних джерел інформації. У дослідженні наведено висновки щодо ефективності методу виявлення порушень та його загальної корисності, з врахуванням Elasticsearch та Python.

Ключовою складовою системи є використання Filebeat для отримання та передачі логів від агентів безпеки Defender у систему Elasticsearch. Rusthound, у свою чергу, виступає як додаткове джерело контекстної інформації, що допомагає збагачувати дані, отримані від Defender, і розширювати їхню інтерпретацію.

Процес виявлення порушень у роботі здійснюється із використанням Python для реалізації скриптів та автоматизації ряду завдань. Для генерації повідомлень про інциденти використано Jinja, що уможливило генерування індивідуалізованих повідомлень.

Використання інтегрованого підходу у дослідженні до аналізу логів забезпечує глибоке та повне розуміння стану безпеки системи. Комбінація Filebeat, Rusthound, Elasticsearch, Python і Jinja створює потужний інструментарій для виявлення, аналізу та реагування на потенційні порушення безпеки, роблячи інформаційні системи більш стійкими та захищеними.

ANNOTATION

Analysis of Intrusion Detection Methods in Information Security Systems// Mykytyshyn Artur Andriyovych // Ternopil National Technical University named after Ivan Puluj, Faculty of Computer and Information Systems and Software Engineering, Department of Cybersecurity, group SBm-61 // Ternopil, 2023 // P. -83, fig. -15, tables -, slides -18.

Keywords: LOG ANALYSIS, IDS, DEFENSE SYSTEM, FILEBEAT, RUSTHOUND, JINJA, ELASTICSEARCH, PYTHON, INCIDENT DETECTION

The qualification work presents a comprehensive approach to detecting security breaches in information systems using log analysis methods and integration of various information sources. The study provides conclusions about the effectiveness of the method of detecting violations and its overall usefulness, taking into account Elasticsearch and Python.

A key component of the system is the use of Filebeat to receive and transfer logs from Defender security agents to Elasticsearch. Rusthound, in turn, acts as an additional source of contextual information that helps enrich the data received from Defender and expand its interpretation.

The process of detecting operational irregularities is carried out using Python to implement scripts and automate a number of tasks. Jinja is used to generate incident notifications, which makes it possible to generate customized messages.

Using an integrated approach to log analysis in the study provides a deep and complete understanding of the system's security status. The combination of Filebeat, Rusthound, Elasticsearch, Python, and Jinja creates a powerful toolkit for detecting, analyzing, and responding to potential security breaches, making information systems more resilient and secure.

ЗМІСТ

ВСТУП

Актуальність теми. За останні роки частота та серйозність кіберзагроз досягли безпрецедентного рівня, що зробило виявлення вторгнень критичним компонентом стратегій інформаційної безпеки. Природа кібернетичних загроз, що постійно змінюється, включаючи передові постійні загрози, експлойти нульового дня та цілеспрямовані атаки, вимагає постійної переоцінки та вдосконалення методів виявлення вторгнень.

З ескалацією кіберзагроз, включаючи розширені атаки та неавторизований доступ, виявлення вторгнень є життєво важливим для захисту інформаційних систем. Зростаюча складність атак у поєднанні з розширенням площі атаки завдяки взаємозв'язку та Інтернету речей підкреслює необхідність постійно оцінювати та вдосконалювати методи виявлення вторгнень. Аналіз існуючих методологій має важливе значення для усунення обмежень, виявлення сильних і слабких сторін і просування розробки активних і стійких систем виявлення вторгнень. Актуальність роботи полягає в удосконалені механізму виявлення аномалій журналів подій та, як наслідок, покращення безпеки та стійкості інформаційних систем перед обличчям кіберзагроз, що розвиваються.

Мета дослідження: дослідити методи виявлення вторгнень, їх переваги та недоліки, реалізувати власну систему виявлення вторгнень.

В роботі поставлено та розв'язано наступні задачі:

- Провести поглиблений аналіз різних інструментів захисту інформації, включаючи брандмауери, антивіруси та віртуальні приватні мережі, окресливши їхні переваги та сфери застосування.;
- Дослідити архітектури систем виявлення вторгнень та їх інтеграції з системами управління інформаційною безпекою.
- Розробити теоретичну базу шляхом аналізу системних журналів, налаштування надсилання журналів та запропонували дизайн власної системи виявлення вторгнень.

– Дослідити механізми виявлення загроз на основі спостерігачів та проаналізувати впровадження систем оповіщення для реагування на інциденти.
– Впровадити практичні рішення, включаючи розробку конвеєра Logstash для аналізу системних журналів, створення модулів спостерігачів для виявлення аномалій та розширення RustHound для покращення аналізу.
– Створити систему сповіщення про інциденти для оповіщення зацікавлених сторін у разі виникнення інцидентів безпеки, що покращує загальний механізм реагування.

Об’єкт дослідження: системи виявлення вторгнень.

Предмет дослідження: методики виявлення вторгнень в інформаційних системах.

Методи дослідження: науковий аналіз, синтез, моделювання.

Наукова новизна отриманих результатів:

– Розроблені спостережувальні модулі для виявлення аномалій. Використовуючи механізми на основі спостерігачів, робота покращує здатність системи виявляти тонкі аномалії, сприяючи більш активній та адаптивній безпеці. На основі даної розробки побудована персоналізована система сповіщення про інциденти. Завдяки механізму сповіщення, робота задовольняє потребу в персоналізованому та ефективному реагуванні на інциденти, забезпечуючи своєчасне та релевантне сповіщення зацікавлених сторін.

Практичне значення одержаних результатів. Практичне значення полягає в розробці вдосконалених протоколів кібербезпеки. Систематичний аналіз традиційних і сучасних інструментів у поєднанні з впровадженням власної системи виявлення вторгнень надає організаціям реальні заходи для захисту своїх інформаційних систем від кіберзагроз.

Апробація. Результати дослідження апробовано на XI науково-технічній

конференції «Інформаційні моделі, системи та технології» у вигляді опублікованих тез (див. Додаток А).

Структура роботи. Робота складається з пояснювальної записки та графічної частини. Пояснювальна записка складається з вступу, 4 розділів, висновків, списку використаної літератури та додатків. Обсяг роботи: пояснювальна записка – 84 арк. формату А4, графічна частина – 18 слайдів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналітичний огляд

У сучасному світі, де величезні обсяги конфіденційних даних передаються мережами, потреба в надійних заходах безпеки є як ніколи актуальною. Системи виявлення вторгнень (IDS) є важливими вартовими цифрового ландшафту, які активно відстежують мережевий трафік і системну активність на предмет виявлення будь-яких ознак зловмисних дій.

Вторгнення в мережу – це будь-яка несанкціонована діяльність в цифровій мережі. Вторгнення в мережу часто пов’язані з крадіжкою цінних мережевих ресурсів і майже завжди ставлять під загрозу безпеку мереж або їх даних [1].

Методика виявлення на основі сигнатур, ветеран сфери ідентифікації вторгнень, покладається на знайому зброю: заздалегідь визначені шаблони і сигнатури відомих атак. Ці сигнатури, ретельно розроблені для виявлення конкретних методик атак, діють як цифрові відбитки пальців, що дозволяє швидко ідентифікувати та нейтралізувати знайомі загрози. Цей метод має перевагу високої точності проти відомих супротивників у поєднанні з низьким рівнем помилкових спрацьовувань, що робить його надійним захисником від усталених векторів атак. Однак його ефективність обмежується постійним розвитком кіберзагроз. Нові атаки, позбавлені характерних ознак, можуть легко прослизнути крізь його захист, роблячи його вразливим перед невідомим. Крім того, постійний розвиток методології атак вимагає безперервного оновлення баз даних сигнатур, що створює значне навантаження на системних адміністраторів.

Виявлення на основі аномалій, новатор у світі виявлення вторгнень, кидає виклик загальновідомим практикам. Замість того, щоб покладатися на заздалегідь визначені шаблони, він використовує більш динамічний підхід. Ретельно вивчаючи "нормальну" поведінку мережі та активність системи, він встановлює базову лінію, відбиток очікуваної активності. Будь-яке значне відхилення від цієї встановленої

норми викликає тривогу, що спонукає до подальшого розслідування. Ця притаманна системі адаптивність дає змогу виявляти аномалії, які раніше не були помічені, зокрема нові варіанти атак і вразливості нульового дня. Однак ця адаптивність має свою ціну. Постійний потік мережевої активності може призвести до високого рівня помилкових спрацьовувань, засипаючи персонал служби безпеки потоком хибних сповіщень. Ретельне налаштування і конфігурація мають вирішальне значення для підтримки балансу між швидкістю реагування і точністю.

Hybrid Detection, дипломат у світі виявлення вторгнень, прагне подолати розрив між двома аналогами. Поєднуючи сильні сторони виявлення на основі сигнатур і аномалій, метод націлений створити єдиний фронт протистояння постійно мінливому ландшафту загроз. Виявлення на основі сигнатур посилює захист від відомих загроз, тоді як виявлення на основі аномалій забезпечує захист від невідомих. Така синергія пропонує переконливе рішення, що забезпечує високу точність і зменшує кількість хибних спрацьовувань. Однак складність, притаманна цьому гібридному підходу, зумовлює необхідність ретельного проектування і впровадження, що вимагає значних технічних знань і обчислювальних ресурсів.

Виявлення на основі специфікацій, пуританин у світі виявлення вторгнень, використовує більш формальний підхід. Він ретельно визначає очікувану поведінку систем і мереж за допомогою формальних специфікацій, схожих на план безпечної роботи. Будь-яке відхилення від цих ретельно розроблених специфікацій позначається як потенційний інцидент безпеки, що вимагає негайного розслідування. Такий суворий підхід забезпечує високоточне виявлення як відомих, так і невідомих вразливостей, пропонуючи високий рівень гарантій для організацій, які прагнуть максимальної безпеки. Однак розробка та підтримка цих комплексних специфікацій може бути складним завданням, що вимагає глибокого розуміння формальних методів і значного технічного досвіду. Крім того, обмежена масштабованість цієї методики може не підходити для всіх середовищ.

Ландшафт кіберзагроз постійно розвивається, вимагаючи постійного

вдосконалення методів виявлення вторгнень. Майбутні дослідження обіцяють більш досконалі алгоритми виявлення аномалій, здатні значно зменшити кількість помилкових спрацьовувань. Штучний інтелект і машинне навчання пропонують потенціал для поліпшення прогнозування загроз і автоматизованого реагування, що ще більше підвищує ефективність систем виявлення вторгнень. Вивчення нових методів виявлення з використанням нових технологій, таких як блокчейн і квантова криптографія, прокладає шлях до ще більш безпечного майбутнього.

1.2 Програмні засоби захисту інформаційних систем

1.2.1 Аналіз та застосування брандмауерів

У цифрову епоху наше життя нерозривно пов’язано із використанням різноманітних технологій. Величезна кількість конфіденційної інформації - від особистої інформації до фінансових даних - зберігається на наших пристроях і в хмарі. Захист цієї інформації від несанкціонованого доступу та зловмисних атак має вирішальне значення, і саме тут на допомогу приходять системи технічного захисту.

Брандмауери - це лідери цифрового світу, які непомітно захищають наші системи від постійної загрози кібератак. Ці цифрові оплоти діють як воротарі, ретельно аналізуючи та фільтруючи вхідний та вихідний трафік, забезпечуючи проходження лише дозволеного зв'язку. Вони є мовчазними охоронцями наших даних, невидимими щитами, що відбивають натиск шкідливого програмного забезпечення та спроби несанкціонованого доступу. Брандмауери використовують різноманітні технічні можливості для захисту комп'ютерних мереж від кіберзагроз. До основних з них належать:

— Фільтрація пакетів — це найпростіший і найпоширеніший метод захисту. Він полягає в аналізі заголовків мережевих пакетів і прийнятті рішення про те, дозволити чи заблокувати їх. Фільтрація пакетів може використовуватися для блокування доступу з певних IP-адрес або портів, а

також для блокування певних типів трафіку, таких як спам або шкідливе програмне забезпечення.

— Аналіз стану з'єднання — це більш складний метод захисту, який полягає в аналізі стану з'єднань між комп'ютерами. Це дозволяє брандмауеру виявляти аномалії в поведінці з'єднань, які можуть бути ознакою атаки.

— Аналіз поведінки програм — це найскладніший метод захисту, який полягає в аналізі поведінки програм, що працюють на комп'ютері. Це дозволяє брандмауеру виявляти програми, які намагаються завдати шкоди комп'ютеру або мережі.

Брандмауери можуть використовувати комбінацію цих методів для забезпечення більш високого рівня захисту. Наприклад, брандмауер може використовувати фільтрацію пакетів для блокування доступу з певних IP-адрес, а також аналіз поведінки програм для виявлення шкідливого програмного забезпечення. Крім того, брандмауери можуть використовуватися для:

— Контролю доступу – брандмауери можуть використовуватися для контролю доступу до певних ресурсів в мережі. Наприклад, брандмауер може бути налаштований для дозволу лише користувачам з певних підрозділів компанії отримувати доступ до корпоративної мережі.

— Ідентифікації та аутентифікації – брандмауери можуть використовуватися для ідентифікації та аутентифікації користувачів, які намагаються отримати доступ до мережі. Це допомагає захистити мережу від несанкціонованого доступу.

— Сегментації мережі – брандмауери можуть використовуватися для сегментації мережі на окремі частини. Це допомагає захистити критичні ресурси мережі від поширення атак.

Брандмауери є важливим компонентом будь-якої комп'ютерної мережі.

Вони допомагають захистити мережу від широкого спектру кіберзагроз, включаючи:

— Шкідливе програмне забезпечення – це програма, призначена для заподіяння шкоди комп'ютеру або мережі. Брандмауери можуть виявляти та блокувати шкідливе програмне забезпечення, яке намагається отримати доступ до мережі.

— Зловмисні атаки – це спроби отримати несанкціонований доступ до комп'ютера або мережі. Брандмауери можуть допомогти відбити зловмисні атаки, блокуючи доступ зловмисникам.

— Інформаційні втрати – це втрата важливої інформації. Брандмауери можуть допомогти запобігти інформаційним втратам, блокуючи доступ зловмисникам до конфіденційної інформації.

Брандмауери є ефективним способом захисту комп'ютерних мереж від кіберзагроз. Однак вони не є панацеєю. Брандмауери слід використовувати в поєднанні з іншими засобами безпеки, такими як антивірусне програмне забезпечення та брандмауери веб-додатків.

1.2.2 Переваги та особливості антивірусів

У величезному цифровому ландшафті, що постійно розвивається, наші системи постійно перебувають під загрозою. Серед нешкідливих на перший погляд файлів і програм ховаються зловмисні об'єкти, відомі як шкідливе програмне забезпечення, які чекають, щоб скористатися вразливими місцями і посіяти хаос. Саме на цьому полі бою одиниць і нулів антивірусне програмне забезпечення стає нашим мовчазним захисником, невтомно працюючи за лаштунками, щоб забезпечити безпеку наших систем.

Антивірусне програмне забезпечення використовує різноманітні технічні можливості для виявлення та видалення шкідливого програмного забезпечення. До основних з них належать:

— Виявлення на основі сигнатур – це найпростіший і найпоширеніший метод виявлення шкідливого програмного забезпечення. Він полягає в

порівнянні файлів у системі з базою даних відомих сигнатур шкідливого програмного забезпечення. Якщо файл містить сигнатуру відомого шкідливого програмного забезпечення, антивірусне програмне забезпечення негайно видаляє його.

— Евристичний аналіз – це більш складний метод виявлення шкідливого програмного забезпечення. Він полягає в аналізі поведінки файлів і програм, щоб виявити підозрілі шаблони, які можуть свідчити про наявність шкідливого програмного забезпечення. Наприклад, антивірусне програмне забезпечення може виявити, що файл намагається записати себе в системні файли або що програма намагається отримати доступ до конфіденційної інформації.

— Аналіз поведінки мережі – це ще один складний метод виявлення шкідливого програмного забезпечення. Він полягає в аналізі мережевого трафіку, щоб виявити підозрілі шаблони, які можуть свідчити про наявність шкідливого програмного забезпечення. Наприклад, антивірусне програмне забезпечення може виявити, що комп'ютер спілкується з невідомим сервером або що комп'ютер генерує значний обсяг мережевого трафіку.

Більшість антивірусних програм використовують комбінацію цих методів для забезпечення більш високого рівня захисту. Наприклад, антивірусне програмне забезпечення може використовувати виявлення на основі сигнатур для виявлення відомого шкідливого програмного забезпечення, а також евристичний аналіз для виявлення невідомого шкідливого програмного забезпечення.

Крім того, антивірусне програмне забезпечення може використовуватися для:

— Захист від фішингу – антивірусне програмне забезпечення може виявляти та блокувати шкідливі веб-сайти, які намагаються обманом змусити користувачів ввести свої особисті дані.

— Захист від руткітів – антивірусне програмне забезпечення може виявляти та видаляти руткіти, які є шкідливим програмним забезпеченням,

яке може проникнути в ядро операційної системи і залишатися непоміченим для антивірусних програм.

— Захист від рекламного програмного забезпечення – антивірусне програмне забезпечення може виявляти та блокувати рекламне програмне забезпечення, яке може переповнювати ваш комп'ютер рекламою.

Антивірусне програмне забезпечення є важливим компонентом будь-якої комп'ютерної системи. Воно допомагає захистити ваш комп'ютер від широкого спектру шкідливого програмного забезпечення, включаючи віруси, троянські коні, шпигунське програмне забезпечення та рекламне програмне забезпечення.

Однак, як замок потребує декількох рівнів захисту, так і надійна стратегія безпеки вимагає більше, ніж просто антивірусне програмне забезпечення. Саме тут у гру вступають VPN.

1.2.3 Технології та застосування віртуальних приватних мереж

По суті, VPN працює шляхом встановлення зашифрованого з'єднання між пристроєм користувача і віддаленим сервером. Це з'єднання інкапсулює всі пакети даних, перетворюючи їх у безпечний, нечитабельний формат. Цей процес, відомий як тунелювання, ефективно захищає онлайн-активність користувача від сторонніх очей, включаючи інтернет-провайдерів, урядові установи і навіть зловмисників. За допомогою VPN можна зашифрувати свій інтернет-трафік і замаскувати свою особу в інтернеті. Це ускладнює стороннім особам відстеження вашої діяльності в інтернеті та крадіжки даних [2].

Кілька технічних механізмів сприяють надійному захисту, який пропонують VPN. Алгоритми шифрування, такі як AES-256, відіграють вирішальну роль у захисті конфіденційності даних, роблячи їх незрозумілими для будь-кого без ключа розшифрування. Крім того, протоколи VPN, включаючи OpenVPN, IKEv2 і L2TP/IPsec, забезпечують надійні та безпечні канали зв'язку для передачі даних.

Застосування VPN виходить далеко за рамки індивідуальних проблем

конфіденційності. Компанії часто використовують VPN для створення захищених з'єднань між своїми співробітниками та корпоративними мережами, що дозволяє віддалено працювати і співпрацювати, забезпечуючи при цьому захист конфіденційних даних. Крім того, VPN виявляються безцінними для обходу географічних обмежень, дозволяючи користувачам отримувати доступ до веб-сайтів і контенту, які можуть бути заблоковані в їхньому місцезнаходженні. Ця функція стає особливо актуальною в країнах із суворою політикою інтернет-цензури.

Крім того, VPN пропонують значні переваги у сфері анонімності в Інтернеті. Маскуючи справжню IP-адресу користувача і замінюючи її на IP-адресу віддаленого сервера, VPN ефективно приховують місцезнаходження користувача та його ідентичність в Інтернеті. Така анонімність може мати вирішальне значення для журналістів, активістів та осіб, які проживають у регіонах з обмеженою свободою вираження поглядів, дозволяючи їм займатися онлайн діяльністю без страху стеження або відплати.

Незважаючи на численні переваги, VPN не позбавлені недоліків. Процес шифрування, хоча і підвищує безпеку, може призвести до підвищення продуктивності, що потенційно може вплинути на швидкість інтернету. Крім того, залежність від віддаленого сервера вимагає певного рівня довіри до VPN-провайдера, оскільки дані користувача в кінцевому підсумку проходять через його інфраструктуру.

Однак, оскільки технологія VPN продовжує розвиватися, ці обмеження постійно усуваються. Удосконалення алгоритмів шифрування та оптимізація апаратного забезпечення мінімізують накладні витрати на продуктивність, а поява безпечних VPN-протоколів і надійних провайдерів підвищує довіру користувачів і безпеку їхніх даних.

Таким чином, VPN є потужним і універсальним інструментом для захисту та анонімізації онлайн-активності. Їх застосування поширюється на різні сфери, від

індивідуальної конфіденційності до безпеки бізнес-мереж, що робить їх важливим компонентом в цифрову епоху. З розвитком технологій і подоланням обмежень VPN відіграватимуть ще більш важливу роль у формуванні майбутнього безпеки і конфіденційності в Інтернеті.

Для підвищення безпеки VPN можна використовувати додаткові засоби захисту, такі як системи виявлення вторгнення (IDS). IDS — це програмне або апаратне забезпечення, яке моніторить мережний трафік на наявність ознак кібератаки. Якщо IDS виявляє потенційну загрозу, вона може сповістити про це системного адміністратора або автоматично блокувати доступ зловмисника..

1.3 Система виявлення вторгнень

1.3.1 Архітектура системи виявлення вторгнень

У лабіринті цифрового світу, де інформація вільно тече, а вразливості ховаються в тінях, захист систем від зловмисних вторгнень є постійною проблемою. Системи виявлення вторгнень (IDS) стають пильними вартовими на цьому полі бою, постійно моніторячи та аналізуючи цифровий ландшафт на наявність слідів підозрілої активності.

Зазвичай, системи IDS мають структуру, яка включає в себе різноманітні компоненти [3], що співпрацюють для забезпечення ефективного функціонування, для прикладу (рис. 1.1):

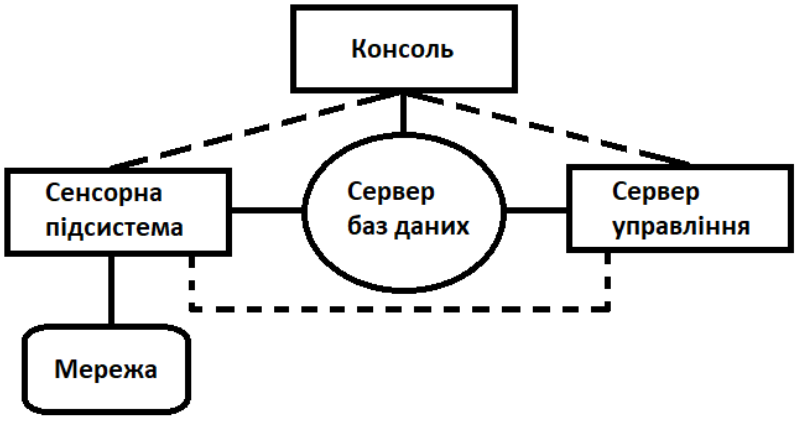


Рисунок 1.1 – Структура СВВ

— Сенсорна підсистема. Ця система контролює та аналізує всю мережеву активність.

— Сервер управління. Цей сервер є центральним пристроєм, який отримує інформацію від сенсорної підсистеми та керує нею. Деякі сервери управління також виконують аналіз інформації про подію і можуть ідентифікувати події, які не може ідентифікувати сенсорна підсистема. Сервери управління доступні як у вигляді пристрою, так і у вигляді програмного продукту. Деякі невеликі IDS не використовують сервери управління, але в більшості вони є. У великих системах виявлення та запобігання вторгнень може бути декілька таких серверів.

— Сервер баз даних. Це сховище інформації про події, яку записали сенсорна підсистема та сервер управління.

— Консоль. Програма, яка надає інтерфейс користувачам та адміністраторам IDS. Таке програмне забезпечення встановлюється на стандартні настільні або портативні комп’ютери. Деякі консолі використовуються лише для адміністрування системи, наприклад для налаштування сенсорів або оновлення програмного забезпечення, в той час як інші консолі використовуються виключно для моніторингу та аналізу. В деяких консолях ці можливості об’єднані.

IDS використовують різні методи виявлення вторгнень. Одні з найпоширеніших методів включають:

— Використовування сигнатур. Цей метод порівнює дані, отримані від сенсорної підсистеми, з базою даних відомих сигнатур атак. Якщо виявлена сигнатура атаки, IDS генерує тривогу.

— Аналіз аномалій. Цей метод шукає відхилення від нормального поведінки системи. Якщо виявлено відхилення, IDS генерує тривогу.

— Аналіз поведінки. Цей метод аналізує послідовність подій, щоб виявити

потенційні атаки.

Також існують набагато більш складні підходи до виявлення неправомірного використання, які можуть використовувати одну сигнатуру для виявлення груп атак [4]. IDS є важливим інструментом безпеки, який може допомогти захистити системи від зловмисних вторгнень. Однак важливо розуміти обмеження IDS.

Існують два основних підходи до аналізу подій для виявлення атак: виявлення неправомірного використання даних, таких як конфіденційні дані або інтелектуальна власність, ресурсів, таких як комп'ютери, мережі або сервери або ж функцій, таких як аутентифікація або авторизація. А також підхід виявлення аномалій. Виявлення зловживання – це аналіз, спрямований на виявлення вже відомої негативної діяльності [6]. Більшість систем використовують цей підхід. З іншого боку, виявлення аномалій шукає незвичайні моделі діяльності, і цей метод менше поширений. Кожен з цих підходів має свої переваги і недоліки. Ефективні системи виявлення вторгнень часто комбінують методи виявлення зловживань з елементами виявлення аномалій.

Простим шляхом атаки або спричинення недієздатності системи є використання "експлойтів" – написаних модулів, що реалізують етапи атаки. Цей метод поширений через доступність великої кількості експлойтів в Інтернеті. Однак такі атаки залишають фіксовані сліди, що полегшує їхнє виявлення.

Системи виявлення зловживань ґрунтуються на створенні шаблонів для атак. Рівень абстракції цих шаблонів може бути простим, як виявлення певних значень у мережевих пакетах, або складним, як відстеження траєкторії системи через небезпечні стани. Хоча ці системи ефективні проти відомих атак, вони стикаються з проблемами при невідомих атаках або відхиленнях від шаблонів, тому потрібна постійна актуалізація бази даних з атаками та їхніми варіаціями.

Детектори аномалій виявляють незвичайну поведінку в мережі, ґрунтуючись на відмінностях від нормальної діяльності. Вони створюють профілі на основі старих даних і визначають аномалії від них. Ці детектори можуть виявляти нові

форми атак, але також можуть породжувати помилкові тривоги через відмінності між нормальною моделлю та реальною поведінкою. Переваги та недоліки кожного підходу варто розглядати в контексті конкретних потреб та умов використання.

1.3.2 Системи управління інформаційною безпекою

Управління інформацією та подіями безпеки (SIEM) – важливий компонент системи кібербезпеки. У динамічному та мінливому середовищі кібербезпеки організації стикаються з постійними викликами щодо захисту своїх систем та мереж від зловмисників та складних атак. Хоча системи IDS відіграють життєво важливу роль у виявленні та попередженні про підозрілу активність, їм часто бракує необхідного контексту та цілісного бачення для забезпечення ефективного виявлення загроз та реагування на інциденти.

У цьому дослідженні розглядається важлива роль рішень для SIEM у зміцненні системи кібербезпеки шляхом підвищення прозорості, сприяння швидшому виявленню загроз та оптимізації зусиль з реагування на інциденти. Сучасний ландшафт загроз представляє все більший виклик для організацій, оскільки кібератаки стають все більш складними, цілеспрямованими і складними для виявлення. Традиційні рішення для забезпечення безпеки, часто обмежені в масштабах і не мають всебічної видимості, намагаються йти в ногу з цими еволюціонуючими загрозами. Рішення для SIEM стають критично важливим компонентом сучасних систем кібербезпеки [9], усуваючи ці обмеження шляхом надання централізованої платформи для збору, аналізу та кореляції даних.

Рішення SIEM пропонують цілий ряд можливостей, які покращують стан кібербезпеки:

— Централізації та аналізу даних з різних джерел: SIEM збирають дані з різних систем безпеки, додатків і мережевих пристроїв, створюючи єдине уявлення про стан безпеки.

— Співставлення подій та виявлення загроз: Аналізуючи та співставляючи

дані з різних джерел, SIEM-системи виявляють приховані патерни та аномалії, що дозволяє виявити потенційні загрози на ранніх стадіях.

— Пріоритетність оповіщень та полегшення реагування на інциденти: SIEM визначають пріоритетність оповіщень на основі серйозності та контексту, дозволяючи командам безпеки зосередити свої зусилля на найбільш критичних загрозах і спростити процедури реагування на інциденти.

— Надання звітності та інформаційних панелей: вбудовані інформаційні панелі та можливості звітування спрощують роботу з дотримання нормативних вимог і надають цінну інформацію про стан безпеки та потенційні вразливості [11].

Інтеграція з системами IDS: хоча IDS ефективно виявляють підозрілу активність на мережевому рівні, їм часто не вистачає контексту, щоб відрізнити справжні загрози від хибних тривог. Інтеграція IDS з SIEM забезпечує необхідний контекст:

— Підвищення точності виявлення загроз: аналізуючи сповіщення IDS разом з іншими даними про безпеку, SIEM відрізняють справжні загрози від хибних спрацьовувань, зменшуючи навантаження на команди безпеки.

— Надання цілісного контексту: SIEM надають комплексне уявлення про події безпеки, що дозволяє командам безпеки розуміти ширший контекст атаки та розробляти більш ефективні стратегії реагування.

Рішення SIEM стають все більш важливими в сучасному ландшафті кібербезпеки. Пропонуючи покращену прозорість, сприяючи швидшому виявленню загроз і оптимізуючи зусилля з реагування на інциденти, SIEM надають організаціям можливість ефективно захищатися від складних кіберзагроз. Існує широкий спектр SIEM-рішень, кожне з яких пропонує унікальні можливості та функції. Деякі з найбільш відомих включають:

— Splunk: Відомий своєю масштабованістю, потужними аналітичними

можливостями та зручним інтерфейсом, Splunk пропонує комплексні функції виявлення загроз та реагування на інциденти. Однак він може бути дорогим і складним у впровадженні для великих розгортань.

— Elastic SIEM: Це рішення з відкритим вихідним кодом може похвалитися потужними можливостями пошуку та аналітики, економічною ефективністю та безперешкодною інтеграцією з різними інструментами безпеки. Його гнучкість дозволяє налаштовувати і розширювати його, що робить його популярним вибором для організацій, які прагнуть розробити власні системи IDS.

— LogRhythm: Ця зручна для користувача платформа пропонує централізоване управління, попередньо створені інформаційні панелі відповідності та автоматизовані робочі процеси реагування на інциденти. Однак вона може бути менш налаштовуваною, ніж інші варіанти, і може вимагати більших витрат при великих розгортаннях.

— Sumo Logic: Ця хмарна SIEM пропонує можливості аналітики та візуалізації в режимі реального часу, що робить її придатною для організацій, які шукають масштабоване та гнучке рішення.

— AlienVault: Цей SIEM з відкритим вихідним кодом фокусується на аналізі загроз і управлінні вразливостями, що робить його цінним активом для проактивного полювання на загрози.

Хоча кожна система виявлення та управління інформацією про події має свої переваги та обмеження, Elastic SIEM відзначається як вибір, що вражає, особливо в контексті виявлення вторгнень. Його унікальні переваги, такі як відкритий вихідний код та масштабованість, забезпечують визначену ефективність. Відкритий вихідний код робить Elastic доступним та економічно вигідним, а можливість масштабування дозволяє адаптуватися до зростаючих обсягів даних та змінюючих потреб у сфері кібербезпеки.

Система володіє потужними засобами пошуку та аналітики, виходячи за межі

інших рішень у цьому сегменті. Це надає командам безпеки зручні інтерфейси та можливість швидко виявляти та розслідувати підозрілу активність завдяки складним запитам.

Elastic легко інтегрується з різними рішеннями систем виявлення вторгнень та іншими інструментами безпеки. Це спрощує консолідацію даних і забезпечує єдиний погляд на події безпеки. Гнучкість та розширюваність Elastic дозволяють організаціям адаптувати його до власних потреб, розробляючи індивідуальні інформаційні панелі, звіти та правила виявлення загроз.

Завдяки використанню мов програмування, зокрема, Python3, який є гнучкою мовою, яка може бути використана для розробки широкого спектру програм. Це дозволяє адаптувати IDS до конкретних потреб свого середовища. Простота використання та зрозумілість мови сприяють швидкій розробці, дозволяючи зосередитися на реалізації складної логіки безпеки, а не на вивченні громіздкого синтаксису. Крім того, Python3 полегшує комунікацію з Elastic Stack, забезпечуючи безперешкодну взаємодію з базою даних Elasticsearch. Ця взаємодія має вирішальне значення для зберігання та пошуку моделей машинного навчання, журналів та іншої важливої інформації. Інтеграція Python3 з Elastic створює цілісну екосистему, де дані безперешкодно перетікають між компонентами, підвищуючи загальну ефективність IDS.

У кваліфікаційній роботі запропоновано новий метод виявлення вторгнень, заснований на сценаріях Python, які виявляють аномалії в системних журналах Elasticsearch. Традиційні системи IDS засновані на правилах, які виявляють відомі атаки. Однак вони не можуть виявляти нові або модифіковані атаки. Системи виявлення вторгнень, засновані на машинному навчанні, можуть виявляти нові та модифіковані атаки, але вони часто вимагають великої кількості даних для навчання. Сценарії з Python, які виявляють аномалії в логах, можуть бути ефективним способом виявлення вторгнень, оскільки вони можуть бути адаптовані до конкретних потреб організації. Даний метод має ряд переваг у порівнянні з

традиційними підходами:

— Він може виявляти як відомі, так і нові атаки.

— Він може бути адаптований до конкретних потреб організації.

— Він може бути реалізований з використанням широкодоступних інструментів.

Для дослідження ефективності даного методу планується провести експерименти на реальних даних з журналів безпеки. Я вважаю, що запропонований мною метод має потенціал для розробки ефективних систем виявлення вторгнень, які можуть захистити організації від кібератак.

2 ТЕОРЕТИЧНА ЧАСТИНА

2.1 Аналіз системних журналів

Системні журнали, також відомі як журнали системних подій, - це набір записів, які документують дії та події, що відбуваються в комп'ютерній системі. Ці журнали слугують літописом поведінки системи, надаючи інформацію про її стан, продуктивність та потенційні загрози безпеці.

Системні журнали можна класифікувати на різні категорії залежно від їхнього призначення та інформації, яку вони фіксують. Деякі поширені типи включають в себе:

— Журнали програм: У цих журналах записуються події, пов'язані з конкретними програмами, такі як запуск програм, помилки та інциденти, пов'язані з безпекою.

— Журнали сервера додатків: Ці журнали відстежують події, пов'язані з серверами додатків, такі як запити до веб-сервера, транзакції з базами даних і автентифікація користувачів.

— Журнали мережі: Ці журнали відстежують мережевий трафік, зокрема вхідні та вихідні з'єднання, втрату пакетів і спроби вторгнення.

— Журнали безпеки: Ці журнали записують події, пов'язані з безпекою, такі як невдалі спроби входу, несанкціонований доступ і зараження шкідливим програмним забезпеченням.

Системні журнали відіграють важливу роль у різних аспектах управління системою та її безпеки. Усунення несправностей та ідентифікація проблем. Системні журнали можна використовувати для виявлення та діагностики системних проблем, таких як конфлікти програмного забезпечення, несправності обладнання та вузькі місця в роботі. Аналізуючи системні журнали, адміністратори можуть відстежувати використання ресурсів, виявляти вузькі місця і оптимізувати конфігурацію системи для підвищення продуктивності. Системні журнали

слугують цінним доказом для аудиту безпеки та розслідування інцидентів. Вони можуть допомогти виявити порушення безпеки, відстежити несанкціонований доступ і визначити джерело зловмисних дій.

Системні журнали є незамінним інструментом для системних адміністраторів, фахівців з безпеки та інженерів з усунення несправностей. Ефективно використовуючи системні журнали, організації можуть отримати цінну інформацію про стан системи, продуктивність і стан безпеки, що дозволить їм активно виявляти і вирішувати потенційні проблеми.

2.2 Конфігурація відправлення системних журналів

Формати та структури реєстрації, що використовуються різними програмними продуктами, часто суттєво відрізняються, що ускладнює ефективний збір та аналіз даних у різних системах. Таке розмаїття підходів до реєстрації даних потребує спеціалізованих інструментів і методологій для полегшення безперешкодної агрегації та аналізу даних.

У дослідженні було вирішено використати журнали реєстрації подій корпорації Майкрософт для навчання розробленої системи виявлення вторгнень. Майкрософт є провідним гравцем в індустрії програмного забезпечення та використовує цілий ряд механізмів реєстрації подій і дій у своїх різноманітних продуктах і послугах. Ці журнали надають цінну інформацію про поведінку системи, її продуктивність та стан безпеки. Однак неоднорідність форматів журналів Microsoft створює проблеми для централізованого моніторингу та аналізу.

Цю проблему можна вирішити за допомогою Filebeat (рис. 2.1), легкого рішення для передачі даних, що включає в себе окремий модуль, спеціально розроблений для отримання даних з продуктів Microsoft. Цей модуль дозволяє збирати та передавати логи з різних служб Microsoft, включаючи defender_atp, m365_defender та dhcp.

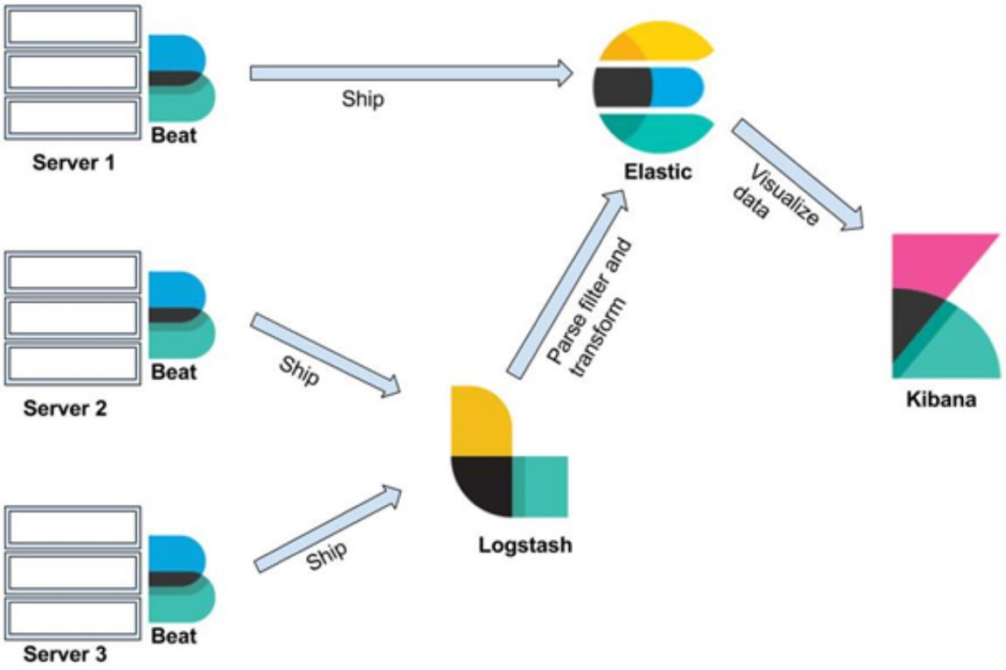


Рисунок 2.1 – Архітектура Filebeat

Файл defender_atp містить журнали, створені Microsoft Defender для кінцевих точок (Microsoft Defender ATP), комплексним рішенням для захисту кінцевих точок. Ці журнали містять детальну інформацію про події безпеки, такі як виявлення шкідливого програмного забезпечення, підозрілі дії та заходи щодо усунення загроз.

Файловий набір m365_defender витягує журнали з Microsoft 365 Defender (Microsoft Threat Protection), уніфікованої платформи безпеки, яка захищає служби та робочі навантаження Microsoft 365. Ці журнали дають уявлення про події безпеки, пов'язані з електронною поштою, хмарними технологіями та ідентичностями, що дає змогу активно виявляти й нейтралізувати потенційні загрози.

Набір файлів dhcp збирає журнали, створені Microsoft DHCP (Dynamic Host Configuration Protocol) - службою, яка динамічно призначає IP-адреси пристроям у мережі. Ці журнали містять інформацію про оренду DHCP, призначення IP-адрес і конфігурації серверів DHCP.

Використовуючи модуль Microsoft у Filebeat, організації можуть ефективно збирати та аналізувати журнали з різних продуктів Microsoft, отримуючи цінну інформацію про стан системи, стан безпеки та потенційні загрози [15]. Цей уніфікований підхід до ведення журналів полегшує комплексний моніторинг і дозволяє активно реагувати на інциденти безпеки.

Процес налаштування Filebeat для модуля Microsoft складається з кількох кроків, починаючи з завантаження та встановлення програмного забезпечення Filebeat. Це можна зробити, перейшовши на веб-сайт Elastic і завантаживши останню версію Filebeat. Після успішного завантаження потрібно перейти до інсталяції Filebeat у системі, де ви збираєтеся збирати логи з джерел Microsoft. Після встановлення Filebeat наступним кроком буде ввімкнення модуля Microsoft у його конфігураційному файлі, який зазвичай знаходиться за шляхом /etc/filebeat/filebeat.yml. Цей модуль дозволяє збирати логи з різних продуктів Microsoft.

Коли модуль Microsoft увімкнений, процес налаштування переходить до визначення шляхів до журналів для збору. У модулі Microsoft визначте набори файлів, що стосуються журналів, які ви хочете зібрати. Для кожного набору файлів вкажіть відповідні шляхи до журналів і будь-які додаткові параметри, необхідні для ефективного збору даних.

Щоб переконатися, що зібрані журнали ефективно надсилаються до потрібного місця призначення, потрібно налаштувати параметри виводу у файлі конфігурації. Зазвичай це включає в себе вказівку кінцевої точки Elasticsearch, де журнали будуть прийматися і зберігатися для подальшого аналізу. Додаткові параметри виводу, такі як рівень журналу і розмір партії, можуть бути налаштовані за необхідності.

```
filebeat.modules:
  - module: microsoft
     enabled: true
     client_id: "YOUR_CLIENT_ID"
     secret: "YOUR_SECRET"
     tenant_id: "YOUR_TENANT_ID"
     scopes: ["https://api.securitycenter.microsoft.com/.default"]
     filesets:
       defender_atp:
         paths:
           - /var/log/microsoft/defender/atp/events.log
       m365_defender:
         paths:
           - /var/log/microsoft/m365defender/events.log
       dhcp:
         paths:
           - /var/log/dhcp/dhcpd.log
```

Рисунок 2.2 – Приклад конфігураційного файлу у Filebeat

Після завершення налаштування можна впевнено зберегти конфігураційний файл Filebeat та запустити службу Filebeat. Це запустить процес збору, і журнали із зазначених продуктів Microsoft будуть надіслані на налаштовану кінцеву точку Elasticsearch. Щоб переконатися в успішному зборі журналів, потрібно перевірити журнали Filebeat та індекси Elasticsearch на наявність отриманих даних журналів. Виконуючи ці кроки, ми можемо ефективно налаштувати Filebeat для збору логів з продуктів Microsoft і легко інтегрувати їх в екосистему Elasticsearch для централізованого зберігання, аналізу та візуалізації.

Наступний крок закладається в тому щоб отримати логи у Logstash, а також надіслати їх на індекс, де можна починати аналіз. Logstash – це універсальний конвеєр обробки даних, відіграє ключову роль у централізації, обробці та аналізі даних журналів з різних джерел, включаючи продукти Microsoft. Його здатність витягувати, трансформувати та завантажувати дані журналів спрощує процес реєстрації, дозволяючи організаціям отримувати цінну інформацію про стан системи, її продуктивність та безпеку.

В основі управління даними в Logstash (рисунок 2.3) лежить індекс -

спеціальна одиниця зберігання даних журналів. Кожен індекс слугує сховищем для категоризованих логів, забезпечуючи організоване зберігання та ефективний пошук. Індекси унікально ідентифікуються за їхніми іменами, що дозволяє чітко класифікувати дані журналів на основі додатків, систем або інших відповідних критеріїв.

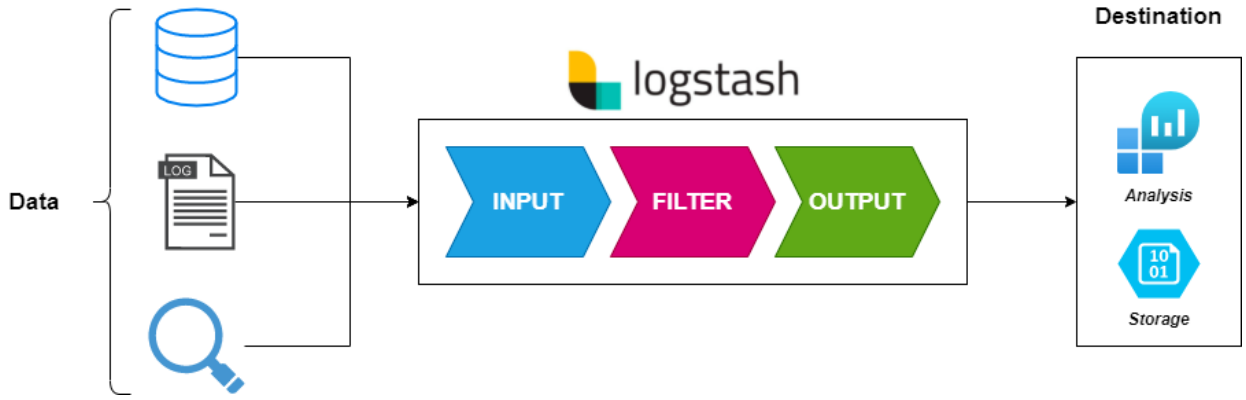


Рисунок 2.3 – Структура Logstash

Щоб ефективно структурувати дані журналів в індексі, Logstash використовує шаблони відображення. Ці шаблони слугують планами, які визначають схему індексу, вказуючи поля, типи даних і зв'язки всередині збережених записів журналу. Шаблон відображення диктує, як кожен запис журналу інтерпретується і структурується в індексі, забезпечуючи послідовне і змістовне представлення даних [3].

Для перетворення та завантаження логів у відповідний індекс Logstash використовує конвеєри – послідовності взаємопов'язаних етапів обробки. Кожен етап виконує певну задачу, наприклад, розбір повідомлень журналу, фільтрацію нерелевантних даних або перетворення даних у потрібний формат.

Етап вхідних даних слугує шлюзом конвеєра, отримуючи лог-дані з різних джерел. Для продуктів Microsoft популярним плагіном є Filebeat, який дозволяє безперешкодно отримувати дані логів з вихідних даних Filebeat.

```
input {
  filebeat {
    port => 5044
    address => "localhost"
  }
}

filter {
  grok {
    patterns_dir => "/path/to/patterns"
    match => "%{TIMESTAMP_ISO8601:timestamp} %{LOGLEVEL:level} %{NOTSPACE:message}"
  }

  mutate {
    add_field => { "source" => "Microsoft" }
  }
}

output {
  elasticsearch {
    hosts => [ "localhost:9200" ]
    index => "microsoft"
  }
}
```

Рисунок 2.4 – Приклад конфігураційного файлу Logstash

Етап фільтрації діє як сито даних, вибірково пропускаючи через конвеєр лише релевантні та значущі записи журналу. На цьому етапі використовуються плагіни-фільтри, такі як grok, для вилучення певних полів з повідомлень журналу та виявлення шаблонів або умов, які визначають, чи слід обробляти запис далі в конвеєрі. В даному випадку використовуються плагіни перетворення, такі як mutate, для маніпулювання полями, додавання нових полів або перетворення типів даних, щоб забезпечити узгодженість і сумісність з шаблоном відображення індексу.

Етап виводу позначає кінцеву точку конвеєра, надсилаючи оброблені дані каротажного журналу до кінцевого пункту призначення. Для Elasticsearch використовується вихідний плагін Elasticsearch, який гарантує, що дані будуть надійно прийняті та збережені у визначеному індексі.

Після впровадження конвеєра Logstash, можна розпочати ефективно централізувати, обробляти та аналізувати дані журналів з продуктів Microsoft.

Інтегрована екосистема Filebeat, Logstash та Kibana дасть змогу отримувати корисну інформацію з даних журналів, що дозволяє активно виявляти та зменшувати потенційні загрози, а також покращувати загальний стан здоров'я та безпеку системи.

2.3 Проектування власної системи виявлення вторгнень

Виявлення аномалій у журналах Microsoft пов'язане з кількома проблемами, які необхідно вирішити для ефективного виявлення та реагування на потенційні загрози:

— Великий обсяг і різноманітність даних: Продукти Microsoft генерують величезні обсяги журналів, що охоплюють різні формати, структури та поля. Такий обсяг і різноманітність створюють значні труднощі для алгоритмів виявлення аномалій, оскільки вони повинні ефективно обробляти і аналізувати великі набори даних, зберігаючи при цьому точність.

— Шум і хибні спрацьовування: Серед величезної кількості регулярних і очікуваних записів у журналі, виявлення справжніх аномалій має вирішальне значення для мінімізації хибних спрацьовувань і зосередження на справжніх загрозах. Хибні спрацьовування можуть перевантажити команди безпеки і призвести до десенсибілізації, що ускладнює виявлення законних вторгнень.

— Загрози, що еволюціонують: Методи виявлення аномалій повинні постійно оновлюватися, щоб адаптуватися до постійно мінливого характеру кіберзагроз. Оскільки з'являються нові загрози і змінюються вектори атак [18], алгоритми виявлення аномалій повинні бути здатні виявляти ці нові шаблони і відповідно адаптувати свої моделі.

Для ефективного вирішення проблем виявлення аномалій у журналах Microsoft можна використовувати комбінацію технічних підходів, зокрема статичні методи, такі як середнє значення, медіана та стандартне відхилення, які значно

відхиляються від нормальних шаблонів. Аналізуючи розподіл логів подій і виявляючи відхилення від норми, статистичні методи можуть позначити потенційні аномалії для подальшого дослідження.

Машинне навчання, а саме алгоритм машинного навчання, такі як контрольоване і неконтрольоване навчання, можуть бути навчені на історичних даних для виявлення аномалій на основі вивчених шаблонів. Алгоритми керованого навчання можна навчати на маркованих даних для виявлення аномалій, які відповідають певним шаблонам або умовам. Алгоритми навчання без нагляду можуть виявляти аномалії без використання мічених даних, що робить їх більш універсальними для виявлення нових і невідомих загроз.

Для Python доступні численні бібліотеки виявлення аномалій, що робить його зручною мовою для розробки рішень для виявлення аномалій у журналах Microsoft. Ці бібліотеки надають готові алгоритми та функції, які можна інтегрувати у власні системи виявлення аномалій. Python пропонує потужне та універсальне середовище для розробки запитів на виявлення аномалій за допомогою Elasticsearch, широко використовуваної пошукової та аналітичної системи. Elasticsearch надає численні варіанти запитів для аналізу даних журналів та виявлення аномалій, включаючи:

— Агрегації: Агрегації дозволяють підсумовувати і групувати дані журналу, що дозволяє ідентифікувати аномальні тенденції або закономірності. Агрегуючи дані журналів на основі певних полів або критеріїв, організації можуть виявляти тенденції, які відхиляються від нормальної поведінки, вказуючи на потенційні аномалії.

— Фільтри: Фільтри дозволяють вибірково знаходити певні записи журналу на основі критеріїв, таких як незвична активність або частота подій. Фільтруючи дані журналу на основі аномалій, виявлених за допомогою агрегацій, організації можуть ще більше вдосконалити свій аналіз і зосередитися на найбільш важливих подіях.

— Інтеграція з машинним навчанням: Elasticsearch інтегрується з бібліотеками машинного навчання, такими як TensorFlow або scikit-learn, що дозволяє розгортати моделі машинного навчання для виявлення аномалій. Ці моделі можна навчати на історичних даних, а потім використовувати для виявлення аномалій в режимі реального часу, забезпечуючи безперервний моніторинг і виявлення загроз.

Щоб ефективно інтегрувати виявлення аномалій в інфраструктуру кібербезпеки, організації можуть розробити спеціальний сервіс для виявлення аномалій. Цей сервіс може бути реалізований на Python з використанням клієнтської бібліотеки Elasticsearch і повинен безперервно відстежувати вхідні дані журналів. При виявленні аномалій сервіс повинен виконувати оповіщення для сповіщення адміністраторів або команд безпеки, що дозволяє оперативно розслідувати і реагувати на них.

Ефективна візуалізація та моніторинг мають вирішальне значення для розуміння інформації, отриманої в результаті виявлення аномалій [20]. Kibana, платформа візуалізації даних, побудована на Elasticsearch, надає повний набір функцій для створення інформаційних панелей, діаграм і графіків, які можна використовувати для відстеження аномалій і виявлення тенденцій з плином часу. Візуалізуючи дані про аномалії, можна отримати цінну інформацію про закономірності та тенденції, які зумовлюють аномальну активність це в свою чергу дозволить вдосконалити моделі виявлення та покращити загальний стан безпеки.

Виявлення аномалій у журналах Microsoft відіграє важливу роль у посиленні кібербезпеки та активному виявленні потенційних загроз. Використовуючи можливості Python, Elasticsearch, машинного навчання та інструментів візуалізації, організації можуть розробляти комплексні рішення для виявлення аномалій, які надають інформацію в режимі реального часу, що дозволяє активно зменшувати ризики та підвищувати стійкість системи.

2.4 Виявлення загроз на основі спостерігачів

Наріжним комнем розроблюваної системи в даній роботі є ретельна побудова еластичних запитів. Ці запити, ретельно розроблені для інкапсуляції різноманітного спектру потенційних загроз, виходять за рамки ролі простих фільтрів. Вони перетворюються на інтелектуальні сутності, які отримали назву "спостерігачі", наділені здатністю ретельно вивчати та інтерпретувати динамічний гобелен системних журналів (рисунок 2.5).

```python
class PhishingDeliveryObserver(ObserverTemplate):
    def __init__(self, source, controller):
        super().__init__(source, controller)
        self.identifier = 403
        self.check_interval = 10
        self.observer_name = 'PhishingDeliveryObserver'
        self.severity_level = 'MEDIUM'
        self.confidence_level = 'MEDIUM'
        self.mitre_techniques = ['T1566', 'T1598']
        self.observation_description = 'Detection of phishing delivery with an allowed policy'
        self.search_query = {
            "query": {
                "bool": {
                    "must": [
                        {
                            "wildcard": {
                                "alert.description": "Phish*"
                            }
                        }
                    ],
                    "must_not": []
                }
            }
        }
```

Рисунок 2.5 – Приклад спостерігача

Уявіть собі сценарій, коли спостерігач, озброєний точно підібраним запитом, перехоплює критичне попередження, згенероване Windows Defender, що сигналізує про наявність раніше невідомої загрози. Можливості спостерігача виходять далеко за межі простого розпізнавання шаблонів. Він призначений для виявлення складних аномалій, що включають несанкціоновану активність користувачів, підозрілі обходи системи та інші ознаки зловмисних намірів, але не обмежуються ними.

Для аналізу журналів Microsoft Defender можна розробити різноманітний набір спостерігачів, кожен з яких пристосований для розпізнавання конкретних

векторів загроз і тактик зловмисників. Окрім PhishingDeliveryObserver, описаного вище, спостерігачі можуть бути розроблені для виявлення аномалій у шаблонах автентифікації користувачів, виявлення підозрілих бічних переміщень у мережі та позначення потенційних випадків ескалації привілеїв. Крім того, спеціалізовані спостерігачі можуть бути націлені на складні методи атак, такі як дампінг облікових даних або використання вразливостей у програмному забезпеченні.

Такий багатогранний підхід дозволяє IDS не лише реагувати на відомі загрози, але й передбачати та адаптуватися до нових сценаріїв атак. У цьому розділі розглядається концептуалізація та впровадження цих різноманітних спостерігачів, підкреслюється їхня колективна роль у підвищенні стійкості інфраструктури кібербезпеки до динамічного ландшафту загроз.

Конкретний спостерігач, позначений наданими атрибутами, називається "PhishingDeliveryObserver". Його роль в рамках загальної системи виявлення вторгнень (IDS) полягає в ретельному вивченні системних журналів на предмет виявлення ознак фішингових атак, спрямованих на користувача. Цей спостерігач, який є частиною ширшого набору спостерігачів, характеризується певними атрибутами та функціями.

— Ідентифікатор (self.identifier): Числовий ідентифікатор, встановлений на 403, слугує унікальним позначенням для PhishingDeliveryObserver в інфраструктурі IDS. Це полегшує систематичну каталогізацію та посилання.

— Інтервал перевірки (self.check_interval): Спостерігач запрограмовано на проведення перевірок через регулярні проміжки часу з попередньо визначеним інтервалом перевірки в 10 одиниць. Такий часовий інтервал забезпечує активний моніторинг системи на предмет потенційних фішингових інцидентів без надмірного навантаження на ресурси.

— Ім'я спостерігача (self.observer_name): Спостерігачеві присвоюється ім'я "PhishingDeliveryObserver", що забезпечує чітку та ідентифіковану мітку для посилання та контекстного розуміння в рамках ширшого спектру

спостерігачів.

— Рівень серйозності (self.severity_level): Рівень серйозності класифікується як "СЕРЕДНІЙ", що вказує на сприйнятий вплив або потенційну шкоду, пов'язану зі спостережуваною подією. Ця класифікація допомагає визначити пріоритети реагування та розподілу ресурсів на основі серйозності виявленої фішингової розсилки.

— Рівень довіри (self.confidence_level): Рівень довіри також оцінюється як "СЕРЕДНІЙ", що відображає ступінь достовірності висновків спостерігача. Цей рейтинг довіри допомагає відрізнити тривоги з високим ступенем достовірності від тих, які можуть потребувати додаткової перевірки.

— MITRE Techniques (self.mitre_techniques): Спостерігач налаштовано на узгодження з конкретними методами MITRE ATT&CK, а саме "T1566" і "T1598". Це відображення встановлює зв'язок між подіями, що спостерігаються, і відомими тактиками, методами і процедурами, які застосовуються супротивниками, покращуючи розвідку загроз і стратегії реагування.

— Опис спостереження (self.observation_description): Текстовий опис "Виявлення фішингової доставки з дозволеною політикою" стисло інкапсулює основну мету спостерігача. Він повідомляє про особливу увагу до виявлення випадків, коли фішингові атаки здійснюються з дотриманням дозволеної політики.

— Пошуковий запит (self.search_query): Серце спостерігача полягає у визначеному пошуковому запиті, вираженому у форматі Elasticsearch Query DSL. Запит сформульовано для ретельного аналізу журналів на наявність оповіщень з описами, що містять термін "Phish*", з використанням пошуку за підстановкою для охоплення варіацій цього терміну. Це дозволяє спостерігачеві виявляти різноманітні випадки фішингових сповіщень.

Під час роботи PhishingDeliveryObserver безперервно відстежує системні

журнали. При виявленні події, коли Windows Defender генерує сповіщення з описом, що вказує на фішингову доставку ('Phish*'), спостерігач запускає реакцію. Ця реакція може включати реєстрацію події, оповіщення системних адміністраторів або запуск заздалегідь визначених заходів, в залежності від налаштованої стратегії реагування в рамках більш широкої архітектури IDS.

Передбачені запити, які слугують основою логіки для цих спостерігачів, є не просто рудиментарними шаблонами пошуку, а скоріше нюансованими конструкціями, які охоплюють безліч сценаріїв загроз. Ці сценарії можуть включати, але не обмежуватися виявленням нових загроз за допомогою сповіщень, що генеруються Windows Defender, проявами аномальних дій користувачів або ознаками складних векторів атак. Спостерігач, подібно до проникливого слідчого, наділений когнітивними здібностями для розпізнавання патернів, що вказують на порушення безпеки.

Однак робота цих спостерігачів виходить за рамки статичного виконання запитів. Для керування розгортанням і виконанням цих спостерігачів у реальному часі передбачено динамічний сервіс Python. Цей сервіс, ретельно розроблений для швидкого реагування, бере на себе роль пильного наглядача, який постійно моніторить системні журнали на предмет збігів, виявлених спостерігачами. Синергія між еластичними запитами і спостерігачами, керованими Python, - це не просто механічний процес; це інтелектуальна взаємодія, симбіоз, в якому запити слугують планом для виявлення, а спостерігачі виконують цей план, уважно стежачи за динамічним ландшафтом загроз, що розвиваються.

Отже, ретельна розробка і впровадження IDS, що характеризується спеціальними запитами, інтелектуальними спостерігачами і оперативним сервісом Python, уособлює наукову прихильність до зміцнення засобів захисту кібербезпеки. Цей підхід виходить за традиційні межі виявлення загроз, запроваджуючи парадигму, в якій система не лише реагує на відомі загрози, але й передбачає та адаптується до нових ризиків. Коли ми заглиблюємося в суть цього стратегічного

підходу, стає очевидним, що забезпечення кібербезпеки – це постійно еволюціонуюча подорож, поєднання інтелекту, адаптивності та активного захисту в цифровій сфері.

2.5 Системи сповіщення для реагування на інциденти

У цьому розділі розглядається важливий аспект систем сповіщення в більш широкому контексті реагування на інциденти. Як невід'ємний компонент архітектури кібербезпеки, системи сповіщення відіграють ключову роль у поширенні своєчасної та дієвої інформації, коли спрацьовує монітор в системі виявлення вторгнень (IDS). У цьому розділі розглядаються принципи побудови, функціональні можливості та стратегії розгортання систем сповіщення, підкреслюється їх важливість для організації ефективного та швидкого реагування на потенційні інциденти кібербезпеки. Розділ має на меті дати уявлення про безперешкодну інтеграцію механізмів сповіщення з загальною системою IDS, сприяючи створенню екосистеми активного реагування на інциденти.

Системи сповіщення про інциденти – це програмне забезпечення, призначене для автоматизації процесу оповіщення відповідних осіб і груп про критичні події. Вони збирають інформацію з різних джерел, включаючи засоби безпеки, системи моніторингу мережі та журнали додатків, а потім аналізують її для виявлення потенційних загроз або інцидентів. Після виявлення інциденту система автоматично запускає заздалегідь визначені сповіщення, інформуючи визначений персонал про характер інциденту, його серйозність і будь-які необхідні негайні дії. Існують різні типи систем сповіщення про інциденти, кожна з яких має свої сильні та слабкі сторони. Типи систем сповіщення про інциденти:

— Системи сповіщення електронною поштою: Сповіщення електронною поштою залишаються наріжним комнем у реагуванні на інциденти, пропонуючи звичний і широко прийнятий канал зв'язку. Автоматизовані

електронні листи можуть швидко передавати важливі деталі про виявлені інциденти визначеним зацікавленим сторонам, сприяючи швидкому реагуванню.

— SMS та текстові сповіщення: SMS-сповіщення забезпечують додатковий рівень оперативності, гарантуючи, що ключовий персонал отримає сповіщення безпосередньо на свої мобільні пристрої. Цей метод ефективний для термінових інцидентів, які вимагають швидкої уваги.

— Інтеграція з інструментами для спільної роботи: Сучасні системи сповіщення про інциденти часто інтегруються з платформами для спільної роботи, такими як Slack або Microsoft Teams. Така інтеграція дозволяє спілкуватися в режимі реального часу, сприяючи співпраці між особами, які реагують на інциденти, і спрощуючи процес прийняття рішень.

— Push-сповіщення в мобільні додатки: Використовуючи спеціальні мобільні додатки, системи сповіщення про інциденти можуть надсилати сповіщення безпосередньо на мобільні пристрої персоналу служби безпеки. Такий підхід гарантує, що працівники служби реагування залишатимуться в курсі подій, навіть перебуваючи за межами своїх робочих місць.

Використання системи повідомлень про інциденти в сфері кібербезпеки має безліч переваг, які сприяють ефективному та оперативному реагуванню на потенційні загрози. Ця інфраструктура забезпечує швидке розповсюдження ключової інформації серед зацікавлених сторін, дозволяючи здійснювати стрімке припинення інцидентів. Технічно, вона забезпечує інтеграцію з різноманітними засобами зв'язку, включаючи традиційну електронну пошту, SMS, а також сучасні платформи спільної роботи, що дозволяє оперативно об'єднувати зусилля команд реагування на інциденти. Такий підхід сприяє зниженню часу простою систем та мінімізації наслідків кіберзагроз, виокремлюючи цю систему як ефективний інструмент для забезпечення кібербезпеки та високого рівня відповідності. Можна виділити наступні основні переваги:

— Своєчасне реагування: Основна технічна перевага полягає у здатності системи сприяти своєчасному реагуванню. Автоматичні сповіщення гарантують, що команди безпеки будуть оперативно поінформовані про потенційні інциденти, що дозволить швидко усунути їх наслідки.

— Скорочення часу простою: Швидке реагування на інциденти за допомогою ефективних систем сповіщення сприяє мінімізації часу простою. Раннє виявлення та втручання допомагають обмежити вплив подій у сфері безпеки, запобігаючи тривалим перебоям в роботі організації.

— Централізована комунікація: З технічної точки зору, системи сповіщення про інциденти забезпечують централізований вузол для поширення інформації серед відповідних зацікавлених сторін. Такий централізований підхід покращує координацію між командами реагування, оптимізуючи вирішення інцидентів.

— Кастомізація та визначення пріоритетів: Багато систем сповіщення про інциденти пропонують технічні можливості для кастомізації та визначення пріоритетів сповіщень. Команди безпеки можуть налаштовувати сповіщення залежно від серйозності інциденту, забезпечуючи тонке реагування на різноманітні події, пов'язані з безпекою.

У даному дослідженні було вирішено вибрати сповіщення електронною поштою, оскільки це безпечний і зручний канал зв'язку для сповіщень про інциденти. Електронна пошта, як традиційна, але надійна платформа, забезпечує надійний засіб передачі критично важливої інформації про потенційні загрози безпеці. Властиві їй функції безпеки, такі як шифрування та автентифікація, сприяють безпечній передачі конфіденційних даних. Крім того, знайомство з електронною поштою та її повсюдність підвищують доступність для призначеного персоналу. Цей вибір узгоджується з виваженим підходом до реагування на інциденти, використовуючи стабільність і широке розповсюдження електронної пошти для забезпечення своєчасного і безпечного поширення важливої інформації,

тим самим зміцнюючи стійкість організації до кібербезпеки.

SilentWatcher – це система, призначена для вилучення даних з Elasticsearch, аналізу їх на предмет потенційних загроз і розсилки сповіщень. Це досягається за допомогою декількох ключових компонентів:

— Екстрактор подій: Цей компонент витягує дані з Elasticsearch на основі визначених користувачем параметрів пошуку.

— EmailComposer: Цей компонент аналізує витягнуті дані і перетворює їх на чіткі та стислі повідомлення електронною поштою.

— Аналізатор загроз: Цей компонент аналізує витягнуті дані на наявність шаблонів і аномалій, виявляючи потенційні загрози.

— Ескалація сповіщень: Цей компонент визначає терміновість і одержувача кожного сповіщення на основі серйозності загрози.

— Генератор звітів: Цей компонент компілює всі отримані дані і створює звіти, щоб надати уявлення про стан і тенденції безпеки організації.

— SilentWatcher постійно навчається і адаптується, що дозволяє йому персоналізувати сповіщення і передбачати нові загрози. Він також може інтегруватися із зовнішніми каналами розвідки загроз для посилення безпеки.

Щоб розширити інформацію в листі, я скористаюся RustHound, потужним інструментом, спеціально розробленим для вилучення інформації про користувачів із середовищ Active Directory. Це дозволить мені зібрати безліч даних про користувачів, включаючи їхні імена, приналежність до доменів, членство в групах тощо. Включивши цю додаткову інформацію в існуючі дані Microsoft, я зможу створити більш повну і детальну картину користувачів в організації.

RustHound, завдяки своїм потужним можливостям, виходить за рамки поверхневої інформації, доступної в журналах Microsoft. Він ретельно витягує деталі, такі як шаблони доступу користувачів, історії входів, асоціації пристроїв і потенційні відхилення від нормальної поведінки. Такий підхід, орієнтований на користувача, значно підвищує глибину аналізу інцидентів, проливаючи світло на

дії, які інакше могли б залишитися непоміченими.

Інтеграція RustHound в систему сповіщення про інциденти є стратегічним кроком вперед в операціях з кібербезпеки. Поєднуючи журнали Microsoft з аналітикою RustHound, орієнтованою на користувача, організації зміцнюють свій захист за допомогою більш тонкого підходу до реагування на інциденти.

```
class EventExtractor:
    def __init__(self, es_client, query):
        self.es_client = es_client
        self.query = query

    def extract_events(self):
        response = self.es_client.search(index="security_events", body=self.query)
        return response["hits"]["hits"]

class EmailComposer:
    def __init__(self, event_hit):
        self.event_hit = event_hit

    def compose_email(self):
        severity = self.event_hit["_source"]["severity"]
        recipient_map = {
            "High": "security@gmail.com",
            "Medium": "security_team@gmail.com",
            "Low": "security_alerts@gmail.com",
        }
        recipient = recipient_map[severity]
        return {
            "subject": f"Security Event Alert: {self.event_hit['_source']['event_type']}",
            "body": f"""
            **Event:** {self.event_hit['_source']['event_type']}
            **Severity:** {self.event_hit['_source']['severity']}
            **Timestamp:** {self.event_hit['_source']['timestamp']}
            **Description:** {self.event_hit['_source']['description']}
            """,
            "recipient": recipient,
        }

class SilentObserver:
    def __init__(self, es_client, email_server):
        self.es_client = es_client
        self.email_server = email_server

    def run(self):
        query = {"query": {"bool": {"must": [{"match": {"severity": "High"}}]}}}

        event_extractor = EventExtractor(self.es_client, query)
        event_hits = event_extractor.extract_events()

        threat_analyzer = ThreatAnalyzer(event_hits)
        threat_analyzer.analyze_threats()

    def send_email(self, email_data):
        server = SMTP(self.email_server)
        server.sendmail(
```

Рисунок 2.6 – Програма системи сповіщень

3 ПРАКТИЧНА ЧАСТИНА

3.1 Розробка Logstash pipeline для аналізу системних журналів

Logstash служить надійним конвеєром обробки даних, який полегшує збір, збагачення та перетворення даних журналу перед надсиланням їх до Elasticsearch. Розуміння тонкощів налаштування Logstash для конкретних випадків використання є важливим для оптимізації робочих процесів аналізу журналів.

Filebeat, невід'ємний компонент нашого складного конвеєра аналізу каротажних даних, слугує швидким і ефективним відправником каротажних даних, призначеним для пересилання та централізації даних каротажних спостережень. Його легка вага не відповідає його потужним можливостям, що робить його незамінною ланкою в ланцюгу обробки даних в нашій системі. У цьому розділі розкривається багатогранна роль Filebeat, проливається світло на його механізми збору даних і безперешкодної передачі даних до рівня обробки Logstash.

У складному ландшафті управління даними колод Filebeat виділяється як стрижень, який організовує збір і транспортування колод. Розташований в авангарді нашої архітектури аналізу даних, Filebeat працює як пильний вартовий, ретельно контролюючи завдання збору та передачі даних журналів.

Характеризуючись своєю легкою конструкцією, Filebeat демонструє вражаючу ефективність в обробці логів. Ця ефективність полягає не лише у споживанні ресурсів, але й у швидкому отриманні даних журналів з різних джерел. Спритність Filebeat свідчить про його здатність виконувати свої обов'язки, не створюючи значного навантаження на загальні ресурси системи.

В основі нашого конвеєра аналізу логів Filebeat бере на себе ключову відповідальність за ініціювання потоку даних. Діючи як початкова контактна точка для даних журналів, він ефективно збирає і передає журнали на наступний рівень обробки - Logstash. Ця передача знаменує собою початок всебічного процесу аналізу, який розплутує хитросплетіння подій, пов'язаних з безпекою.

Ключове розширення можливостей Filebeat реалізовано завдяки безшовній інтеграції модулів Microsoft Defender. Ця інтеграція піднімає Filebeat на вищий рівень, надаючи йому можливість фіксувати нюанси та детальні події, пов'язані з безпекою, що генеруються Microsoft Defender.

Включення модулів Microsoft Defender робить Filebeat більш чутливим до нюансів безпеки. Це призводить до захоплення детальних подій безпеки, забезпечуючи детальне розуміння операційних аспектів системи. Події, пов'язані з виявленням загроз, вразливостей системи та інших параметрів безпеки, всебічно реєструються, створюючи надійну основу для подальшого аналізу.

Забезпечення безперебійного потоку збагачених логів до конвеєра Logstash залежить від ретельної конфігурації. Правильне налаштування Filebeat, особливо в поєднанні з модулями Microsoft Defender, вимагає тонкого розуміння як архітектури системи, так і специфічних характеристик даних журналу. Добре налаштований екземпляр Filebeat стає каналом, через який проходить велика кількість інформації про безпеку, створюючи основу для глибокого аналізу на наступних етапах конвеєра обробки логів.

Таким чином, Filebeat стає ключовим компонентом, який не тільки ефективно керує збором і передачею даних журналів, але й, завдяки інтеграції з модулями Microsoft Defender, бере на себе розширену роль у захопленні та передачі детальної інформації про події, пов'язані з безпекою. Його правильна конфігурація є основою для безперебійного та збагаченого потоку логів до конвеєра Logstash, що закладає основу для надійного аналізу безпеки.

Налаштування Filebeat для оптимальної інтеграції з модулями Microsoft Defender і перенаправлення отриманих збагачених журналів безпеки до Logstash - це складний процес, який вимагає ретельної уваги до деталей. Початок конфігурації йде з встановлення Filebeat на цільовій системі. Потрібно виконати наступну команду для встановлення Filebeat:

— sudo service filebeat start

Далі потрібно перейти до конфігураційного файлу Filebeat (filebeat.yml), щоб встановити Logstash як місце призначення виводу. Я вказую свій локальний сервер і порт Logstash:

— output.logstash:

— hosts: ["127.0.01:5044"]

Дальше потрібно використати розширені можливості Microsoft Defender, увімкнувши відповідний модуль Filebeat. Це знаходиться у файлі microsoft.yml у каталозі модулів Filebeat, а потрібно саме для параметра enabled поставити значення true:

— module: microsoft

— defender:

— enabled: true

Наступний крок це налаштування параметрів модуля Microsoft Defender відповідно до особливостей системи. Потрібно знову перейти в файл microsoft.yml і налаштувати такі параметри, як var.logs, щоб вказати розташування відповідних файлів журналів, а також вказати облікові дані.

Звичайно ж, після цих кроків достатньо запустити службу Filebeat, щоб розпочати процес збору та передачі логів, а також перевірити стан Filebeat, щоб переконатися, що він працює без помилок:

— sudo service filebeat start

— sudo service filebeat status

У сфері аналізу та централізованого управління журналами Logstash виділяється як потужний і універсальний інструмент, який полегшує агрегацію, трансформацію та збагачення даних журналів. В основі Logstash лежить концепція конвеєра - послідовності етапів, через які проходять дані, піддаючись різним діям обробки, перш ніж досягти місця призначення. У Kibana створення конвеєра Logstash передбачає визначення складного потоку каротажних даних, вказуючи, як вони трансформуються і доповнюються для досягнення бажаних цілей аналізу.

Одним з найважливіших аспектів конвеєрів Logstash є шаблон відображення. Цей шаблон диктує структуру даних, які будуть індексуватися в Elasticsearch, забезпечуючи однорідність і узгодженість збережених записів журналу. Крім того, політика життєвого циклу регулює управління індексами, диктуючи такі дії, як перекидання, видалення тощо, щоб підтримувати організоване і ефективне сховище даних журналів.

Створення конвеєра Logstash в Kibana передбачає ретельне врахування цих елементів. Визначення послідовності фільтрів, збагачень і перетворень в конвеєрі дозволяє користувачам адаптувати потік обробки логів до конкретних випадків використання. Візуальне представлення в Kibana забезпечує інтуїтивно зрозумілий інтерфейс для створення цих конфігурацій, роблячи процес доступним і зручним для користувача.

Виходячи за межі конвеєра, Logstash підтримує безліч вхідних і вихідних даних. У контексті дослідницької роботи використання Beats як механізму введення дозволяє Logstash ефективно отримувати дані з різних джерел, забезпечуючи гнучкість і адаптивність до різних форматів журналів. На виході, надсилання оброблених даних журналу до Elasticsearch встановлює безперешкодний зв'язок з можливостями індексації та зберігання цієї надійної пошуково-аналітичної системи.

У самому конвеєрі Logstash є місце для кастомізації та вдосконалення. Різні фільтри можна застосовувати для аналізу, розбиття та маніпулювання записами журналів, гарантуючи, що отримані дані будуть відповідати бажаному формату для всебічного аналізу. Крім того, Logstash дозволяє інтегрувати плагіни, розширюючи його функціональність відповідно до конкретних потреб. Така розширюваність робить Logstash універсальним інструментом, здатним розвиватися разом з динамічними вимогами до обробки та аналізу логів у складних середовищах.

На рисунку 3.1 зображено Logstash конвеєр, він починається з визначення джерела вхідних даних, яке, як очікується, отримуватиме логи від Filebeat через

порт 5044. Плагін вводу beats встановлює канал зв'язку, що дозволяє Logstash безперешкодно отримувати логи з різних джерел.

```
input {
  beats {
    port => 5044
  }
}

filter {
  if [event_data][log_name] == "Microsoft-Windows-Windows Defender/Operational" {
    grok {
      match => { "message" => ["%{TIMESTAMP_ISO8601:timestamp} %{LOGLEVEL:log_level} %{GREEDYDATA:log_message}"] }
    }

    grok {
      match => {
        "log_message" => [
          "Scan Started: Scan Type: %{DATA:scan_type}, Engine Version: %{DATA:engine_version}, User: %{DATA:user}",
          "Scan Ended: Scan Type: %{DATA:scan_type}, Engine Version: %{DATA:engine_version}, User: %{DATA:user}, Scan Result: %{DATA
        ]
      }
    }

    if [user_ip] {
      geoip {
        source => "user_ip"
      }
    }

    date {
      match => ["timestamp", "ISO8601"]
    }

  }
}

output {
  elasticsearch {
    hosts => ["127.0.0.1:9200"]
    index => "microsoft_defender_logs-%{+YYYY.MM.dd}"
  }
```

Рисунок 3.1 – Logstash конвеєр

Наступна секція фільтра інкапсулює ряд кроків обробки, адаптованих спеціально для логів Microsoft Defender. Початковий умовний оператор, if [data_data][log_name] == "Microsoft-Windows-Windows Defender/Operational", фільтрує вхідні події, щоб зосередитися виключно на журналах, створених Microsoft Defender.

Фільтр grok використовується для препарування повідомлень журналу, витягуючи важливі фрагменти інформації, такі як мітка часу, рівень журналу і первинне повідомлення журналу. Цей крок покращує структуру журналу, роблячи його більш придатним для подальшого аналізу

Наступний фільтр grok застосовується для подальшого аналізу конкретних

деталей, пов'язаних з процесом сканування. Сюди входить вилучення такої інформації, як тип сканування, версія движка, користувач і результат сканування. Конфігурація є гнучкою і пристосована до різних сценаріїв сканування.

Додатковий крок збагачення GeoIP вводиться для IP-адрес користувачів. Якщо присутнє поле user_ip, застосовується фільтр geoip, який зіставляє IP-адреси з географічним розташуванням. Це забезпечує додатковий контекст для записів журналу, що потенційно може допомогти в аналізі загроз.

Фільтр дати забезпечує точний аналіз мітки часу на основі формату ISO8601. Цей крок має вирішальне значення для вирівнювання подій у хронологічному порядку та полегшення аналізу на основі часу.

Умовні дії вводяться на основі результату сканування. У цьому випадку, якщо в полі scan_result вказано "Виявлено зловмисне програмне забезпечення", можна вжити додаткових заходів, наприклад, запустити оповіщення або виконати заздалегідь визначені реакції.

На останньому кроці оброблені журнали надсилаються до Elasticsearch, вказуючи хост та індекс, де будуть зберігатися збагачені дані. Ця розширена конфігурація конвеєра Logstash призначена для комплексної обробки журналів Microsoft Defender. Кожен крок фільтрації та обробки сприяє покращенню структури, збагаченню даних та наданню індивідуальних відповідей на основі специфічних характеристик записів журналу Microsoft Defender. Це ілюстрація того, як Logstash можна налаштувати для вирішення складних завдань обробки журналів у витончений і адаптивний спосіб.

На завершення, Logstash слугує ключовим компонентом у стеку ELK (Elasticsearch, Logstash, Kibana), пропонуючи надійне рішення для управління журнальними даними. Створення та налаштування конвеєрів Logstash в Kibana дозволяє користувачам адаптувати процес обробки логів, оптимізуючи його для своїх конкретних випадків використання. Інтеграція вхідних даних, таких як Beats, і вихідних даних, таких як Elasticsearch, створює цілісний робочий процес, в той час

як розширюваність Logstash забезпечує адаптивність до мінливих потреб в обробці логів.

3.2 Розробка модулів спостерігачів для виявлення аномалій

Щоб осягнути широту і глибину підходу, необхідно зрозуміти основоположну концепцію спостерігачів. Ці спеціалізовані компоненти діють як пильні вартові, ретельно вивчаючи постійно зростаючі обсяги журналів Elasticsearch. У розділі також підкреслюється ключова роль, яку відіграє Elasticsearch, що слугує не лише сховищем для журналів, але й динамічною і потужною пошуковою системою, яка полегшує ефективний пошук.

Спостерігачі, як зазначено в вище, є спеціалізованими організаціями, що мають єдине завдання - ретельно вивчати та інтерпретувати об'ємні журнали, що містяться в Elasticsearch. Вкрай важливо оцінити симбіотичний зв'язок, який існує між цими спостерігачами і Elasticsearch, не просто як сховищем даних, а як динамічною і потужною пошуковою системою, що сприяє підвищенню ефективності наших зусиль з виявлення загроз.

Наша подорож починається з відшарування шарів класу PhishingDeliveryObserver, мікросвіту архітектури спостерігача. Ми ретельно вивчаємо кожен атрибут - від ідентифікатора, check_interval до observer_name - з'ясовуючи їхню роль в унікальній класифікації та посиланнях на спостерігачів у ширшій інфраструктурі IDS.

При заглибленні в архітектуру, акцент робиться на конфігурованості, притаманній класу спостерігачів. Гнучкість налаштування кожного спостерігача до різних сценаріїв загроз стає очевидною, прикладом чого є спеціалізований фокус на виявленні інцидентів доставки фішингових атак.

Виходячи за рамки індивідуальних характеристик PhishingDeliveryObserver, у розділі представлено набір класів спостерігачів, яких загалом 10. Кожен клас

являє собою модульний і масштабований об'єкт, створений з точністю для вирішення різних аспектів кібербезпеки. Різноманітність цих класів підкреслює адаптивність системи спостерігачів до різноманітних ландшафтів загроз.

Метод run - це серце Observer, що керує його виконанням та управлінням оповіщеннями. Цей метод інкапсулює основну логіку, яка керує взаємодією Observer з Elasticsearch, виявленням сигналів та обробкою потенційних аномалій. Давайте заглибимося в тонкощі методу запуску, досліджуючи його структуру, ключові функції та загальний потік.

У широкому ландшафті методу прогону нескінченний цикл виступає фундаментальною конструкцією, що огортає функціональність Спостерігача в парадигмі безперервного виконання, забезпечуючи постійну пильність і чуйність. Цей цикл організовує циклічне виконання основних завдань Спостерігача, що супроводжується періодичними перевірками на наявність нових сигналів і аномалій, що є ключовим аспектом його всеосяжної місії.

Оскільки метод прогону починає кожну ітерацію, ключовим аспектом його роботи є ретельне відстеження часу. Використовуючи функцію datetime.now(), метод старанно фіксує поточну позначку часу, встановлюючи часову точку відліку, необхідну для обчислення часу, що минув під час виконання Спостерігача. Ця часова обізнаність сприяє здатності методу синхронізувати свою діяльність і підтримувати цілісну часову структуру.

Метод запуску розширює сферу свого впливу на управління клієнтами та джерелами за допомогою вкладених циклів, ретельно перевіряючи кожного підписаного клієнта та пов'язані з ним джерела. Ця дворівнева стратегія ітерацій надає Спостерігачеві всеосяжну сферу застосування, дозволяючи йому ретельно вивчати і взаємодіяти з кожною парою клієнт-джерело в систематичний спосіб.

У складній структурі методу запуску особливу увагу привертає робота зі спостережниками, що запускаються на вимогу. Така конфігурація надає спостерігачеві можливість розпізнавати конкретні джерела і клієнти, що

запускаються на вимогу, надаючи процесу виконання індивідуальний і селективний характер. Користувачі, озброєні гнучкими можливостями запускати Observer на власний розсуд, мають можливість тонко контролювати періодичність роботи цієї пильної сутності.

Метод прогону, з його багатогранними обов'язками, зосереджує свою увагу на перевірці джерел і клієнтів. Метод здійснює критичні перевірки, щоб визначити, чи вимкнено Спостерігач для конкретного клієнта, чи, навпаки, увімкнено на основі динамічної взаємодії конфігурацій білих і чорних списків. Ці перевірки формують невід'ємний рівень безпеки, окреслюючи межі роботи Observer.

У складній фільтрації клієнтів і білих списків метод запуску організовує створення запиту (bq) для кожного клієнта. Цей запит, стрижень механізму фільтрації даних Спостерігача, геніально поєднує в собі мозаїку критеріїв. Глобальні білі списки, білі списки конкретних клієнтів і чорні списки об'єднуються в запиті, разом формуючи проникливу лінзу "Спостерігача" для просіювання даних.

Метод run, який зображений на рисунку 3.2, у зеніті свого операційного крещендо, розгортає свої основні завдання для кожного дуету клієнта і джерела.

Ключовий метод run_client, подібно до палички маестро, спрямовує взаємодію Спостерігача з Elasticsearch. Він організовує тонкий балет виявлення збігів, розкриття тонких закономірностей аномалій і вправну роботу з оповіщеннями.

Після завершення своїх різноманітних завдань метод прогону граціозно переходить у сплячку, викликану інтервалами. Цей тимчасовий перепочинок, продиктований налаштованим інтервалом, символізує момент відпочинку перед тим, як метод відновить свою циклічну подорож. У спеціалізованій сфері спостерігачів машинного навчання (ML) розгортається додатковий рівень складності. Метод, усвідомлюючи, що він є спостерігачем ML, робить активний крок, щоб очистити кеш аномалій ML. Цей стратегічний крок призводить до оновлення даних про аномалії, надаючи Спостерігачеві оновлену інформацію для

наступних ітерацій.

```
def run(self):
    """
    Run Observer loop
    """
    # Lambda function used to check if current Observer is disabled for a specified source
    is_source_disabled = lambda cs, source, wname: cs.get("name") == source and wname in cs.get("disabled", ())
    # Keep loop alive
    while True:
        time_begin = datetime.now()

        # For each client subscribed for that sources check for alerts
        for client in self.clients():
            # For each client source
            for source in self.sources:
                # Check if Observer is on demand
                if self.on_demand:
                    # Check if current source is in client on demand sources
                    if source not in self._cr.configs["clients"].get(client, {}).get("ondemand_Observers", ()):
                        self._logger.info("Skipped '%s' on demand Observer for client '%s' and source '%s'" % (
                            self.name, client, source
                        ))
                        continue

                    # Check if Observer is under current client on demand source
                    if self.name not in self._cr.configs["clients"][client]["ondemand_Observers"][source]:
                        self._logger.info("Skipped '%s' on demand Observer for client '%s' and source '%s'" % (
                            self.name, client, source
                        ))
                        continue

                self._logger.info("Running source '%s' for client '%s'" % (source, client))
                # Check if Observer is disable for current client
                # Iterate on each client source and chech if current Observer is disabled for current client
                if any(is_source_disabled(cs, source, self.name) for cs in
                       self._cr.get("clients", client, "sources")):
                    self._logger.info(
                        "Observer '%s' disabled for source '%s' and client '%s'" % (self.name, source, client))
                    continue

                # Add client
```

Рисунок 3.2 – Програма методу run

Метод run_client слугує операційним ядром у методі run, відповідальним за виконання завдань Observer для вказаної пари клієнт-джерело. Його багатогранні обов'язки охоплюють взаємодію з Elasticsearch, виявлення збігів і розумну обробку отриманих сповіщень.

Метод починається зі створення запиту (bq), адаптованого до конкретного клієнта. Цей запит включає синтез глобальних білих списків, специфічних для клієнта білих списків і чорних списків. Завдяки ретельній інтеграції цих критеріїв

метод гарантує, що подальша взаємодія з Elasticsearch відповідатиме визначеним обмеженням, що полегшує процес фільтрації даних з урахуванням контексту.

Після побудови запиту метод run_client проникає в саме серце взаємодії з Elasticsearch. Він отримує індекс, що відповідає поточному джерелу, і переходить до виконання методу Observer_hits. Цей ключовий крок полягає у виявленні сигналів або збігів в Elasticsearch на основі сформульованого запиту. Потім метод динамічно адаптує своє виконання на основі отриманих результатів.

У разі виявлення збігів, що вказують на потенційні події безпеки, метод організовує функцію upload_alerts. Ця функція інкапсулює процес завантаження цих збігів до Elasticsearch, забезпечуючи структуроване представлення сповіщень. Метод завершується реєстрацією відповідної інформації, що свідчить про успішне виконання завдань Observer для вказаного клієнта і джерела.

Метод detection_hits втілює здатність Observer розпізнавати сигнали або виявлення в Elasticsearch, спеціально розроблений для сигналів SIEM. Слугуючи стрижнем для ідентифікації подій безпеки, цей метод з точністю переміщується в області Elasticsearch.

Метод починається з побудови BoolQuery, який інкапсулює параметри запиту, необхідні для детального пошуку. Він включає такі специфікації, як клієнт, ім'я виявлення та модуль джерела, узгоджуючи запит з характерними особливостями сигналів SIEM. Маючи на руках запит, метод detection_hits взаємодіє з Elasticsearch, використовуючи контролер для доступу до документів з індексу сигналів SIEM. Збіги, отримані за допомогою цього запиту, представляють потенційні інциденти безпеки, що відповідають визначеним критеріям.

Цей метод, з його здатністю точно визначати сигнали SIEM, відіграє ключову роль у виконанні головної місії Observer - виявлення загроз і реагування на них. Він діє як проникливе око, скануючи індекси Elasticsearch, щоб виявити аномалії, що вказують на загрози безпеці.

Представлений шаблон, що інкапсулює методи run, run_client і detection_hits,

забезпечує універсальну основу для створення безлічі спостерігачів, пристосованих до різних сценаріїв атак. Використовуючи цей шаблон, можна створювати спостерігачі, пристосовані до конкретних типів атак, кожен з яких характеризується унікальними параметрами запитів, критеріями перевірки та протоколами взаємодії з Elasticsearch.

Наприклад, в контексті журналів Windows Defender, Observer можна ретельно налаштувати для виявлення і реагування на різні типи кіберзагроз. Незалежно від того, чи це виявлення шаблонів, пов'язаних зі шкідливим програмним забезпеченням, підозрілою діяльністю або вразливостями системи, цей шаблон пропонує гнучкість для адаптації поведінки Observer відповідно до нюансів журналів Windows Defender.

Озброєний спеціальною конфігурацією, Observer стає вартовим, налаштованим на тонкощі журналів, що генеруються Windows Defender. Завдяки стратегічним конструкціям запитів, перевіркам для конкретних клієнтів і розумній обробці збігів і аномалій, Observer стає потужним інструментом для зміцнення кібербезпеки.

По суті, шаблон слугує планом для створення надійної екосистеми Observer, кожен з яких пристосований до особливостей окремих векторів атак. Такий модульний підхід дозволяє фахівцям з кібербезпеки розгортати різноманітні набори спостерігачів, підвищуючи їхню здатність виявляти та реагувати на різноманітний спектр кіберзагроз у ландшафті журналів Windows Defender.

3.3 Розробка розширення RustHound для посилення аналізу

У домену Active Directory, де користувачі, групи, комп'ютери та організаційні одиниці переплітаються в танці доступу та привілеїв, на сцену виходить мовчазний спостерігач. RustHound, крос-платформний інструмент, викуваний в горнилі мови програмування Rust, переміщається в цьому лабіринті з точністю наукового

інструменту.

Не обтяжений обмеженнями операційних систем, RustHound перетинає межі платформ, переходячи між Windows, Linux та macOS з витонченістю квантової частинки. Його непомітна природа, що нагадує субатомне нейтрино, дозволяє йому прослизати повз антивірусний захист, непомітно збираючи розвідувальні дані.

Подібно до мас-спектрометра, що розтинає молекулу, RustHound фрагментує домен на складові частини. Користувачі, групи, комп'ютери та організаційні одиниці ретельно препаруються, а їхні взаємозв'язки ретельно картографуються. Привілейоване членство, вкладені групи та "сплячі" службові акаунти виявляються розкопаними, а їхні секрети - оголеними.

Ці необроблені дані, колись розпорошені і непрозорі, потім кристалізуються у структуровану мову JSON, універсальний формат, зрозумілий BloodHound, картографу домену. Завдяки гострому оку BloodHound з'являється карта, що світиться, розкриваючи приховані шляхи домену, його вразливі місця, що видніються, наче пульсари на космічному полотні.

Озброївшись новими знаннями, червоні командири стають схожими на астрофізиків, які досліджують гравітаційні аномалії домену. Вони виявляють найсоковитіші цілі, надто привілейовані групи та вразливі акаунти, що дозріли для експлуатації. Але репертуар RustHound виходить за рамки простого спостереження. Він перетворюється на віртуальний прискорювач частинок, імітуючи реальні атаки, бомбардуючи захист домену цільовими зондами.

Неправильні конфігурації та слабкі місця, темна матерія домену, виявляються на світлі. RustHound стає горнилом, перевіряючи міцність системи безпеки, виявляючи вразливості до того, як ними скористаються реальні зловмисники. Але RustHound - не самотній об'єкт. Він процвітає в динамічній екосистемі, спільноті розробників і фахівців з безпеки, які постійно вдосконалюють його можливості, додаючи нові функції та можливості. Цей дух співпраці забезпечує адаптивність RustHound, дозволяючи йому розвиватися разом з постійно

мінливим ландшафтом загроз.

На закінчення, RustHound - це не просто інструмент; це наукова методологія, скальпель для препарування складної анатомії домену Active Directory. Він дає можливість "червоним командам" спостерігати, аналізувати і використовувати, в кінцевому підсумку зміцнюючи систему безпеки, виявляючи вразливості до того, як вони будуть використані на озброєння. Оскільки цифровий ландшафт продовжує розвиватися, RustHound з його платформо-діагностичним характером, непомітною роботою і розробкою, керованою спільнотою, готовий залишатися кращим інструментом як для червоних команд, так і для професіоналів з безпеки.

На рисунку 3.3 продемонстровано метод ініціалізації класу AzureGraphApi. Цей сценарій відіграє життєво важливу роль у ширшому ландшафті управління організаційними даними. Взаємодіючи з Azure Graph API, він використовує надійну інфраструктуру Microsoft для вилучення детальної інформації про користувачів, групи, журнали входів і журнали виявлення ризиків. Водночас він використовує Rusthound, додаткове джерело даних, для отримання детальної інформації, збагачуючи набір даних контекстними деталями. Зібрані дані потім ретельно зберігаються в базі даних CouchDB, забезпечуючи безпечне та організоване сховище, яке слугує основою для подальшого аналізу.

На додаток до можливостей збору даних, скрипт узгоджується з перспективною стратегією шляхом інтеграції з Elasticsearch. Ця інтеграція переводить зібрані дані у сферу розширеної аналітики, надаючи потужні пошукові можливості та полегшуючи складні запити до даних.

У цій доповіді буде висвітлено внутрішню роботу скрипту, розглянуто його компоненти та функції, а також продемонстровано його значення в сучасному ландшафті управління організаційними даними та кібербезпеки.

```
Artur Mykytyshyn +1 *
def __init__(
        self,
        organization_code: str = "",
        logstash_log=None,
        debug_log=None
):
    """
    Initializes a new GraphAPI object.

    Args:
        organization_code: The code of the organization to interact with.
        logstash_log: A logger to use for logging to Logstash, if any.
        debug_log: A logger to use for debug logging, if any.
    """

    self.customer = organization_code
    self.debug_logger = debug_log
    self.logstash_logger = logstash_log
    self.session = requests.Session()
    self.session.verify = True
    self.couch_db = CouchReader(
        db_name=f"www-{self.customer}",
        host=os.getenv("HOST_COUCH"),
        port=int(os.getenv("PORT_COUCH")),
        ssl=True,
        auth=(
            os.getenv("USER_COUCH"),
            os.getenv("PASSWORD_COUCH"),
        )
    )
    self.access_token = self.generate_token()
    self.session.headers.update(
        {
            "Authorization": "Bearer {}".format(self.access_token),
            "Content-Type": "application/json",
        }
    )
    self.rusthound_users, self.rusthound_groups = self.get_rusthound_data()

    # interesting containers, which stands for keeping processed SIDs of the users
    self.processed_users_sids = list()
    self.processes_groups_sids = list()
```

Рисунок 3.3 – Код програми – метод ініціалізації

Сценарій починає свою подорож з важливого етапу ініціалізації, налаштування основних компонентів, які є невід'ємною частиною подальшої безпечної та надійної обробки даних. Сюди входить створення логів для налагодження та реєстрації в Logstash, ініціювання безпечного HTTP-сеансу з Azure Graph API, а також встановлення з'єднання з базою даних CouchDB. Ці компоненти разом створюють основу для надійного і безпечного середовища обробки даних.

На цьому етапі скрипт імпортує необхідні бібліотеки, такі як json для обробки даних JSON, os для взаємодії з операційною системою та requests для створення HTTP-запитів. Крім того, включаються можливості асинхронної обробки (ThreadPoolExecutor) і зовнішні конфігурації з файлів (dotenv, settings), що забезпечує універсальну і адаптивну архітектуру скрипта.

Центральним елементом архітектури скрипта є визначення класу AzureGraphAPI, який інкапсулює основні функції, необхідні для взаємодії з Azure Graph API. Цей клас включає методи для генерації токенів автентифікації, отримання журналів підписання та оновлення документів у базі даних CouchDB. Структура класу розроблена для модульності, масштабованості та простоти обслуговування.

Метод класу generate_token (рисунок 3.4) виділяється як ключовий компонент, що відповідає за отримання токену доступу з Microsoft Graph, що є вирішальним кроком у процесі автентифікації. Метод get_signing_logs та інші є невід'ємною частиною подальшого процесу збору даних.

Щоб адаптуватися до різних організаційних контекстів, сценарій покладається на властивості конфігурації, що зберігаються в базі даних CouchDB. Ці властивості містять таку важливу інформацію, як ідентифікатор орендаря Azure AD, ідентифікатор клієнта, секрет клієнта та інші глобальні змінні. Під час ініціалізації скрипт отримує ці властивості, забезпечуючи динамічний і настроюваний підхід до пошуку даних.

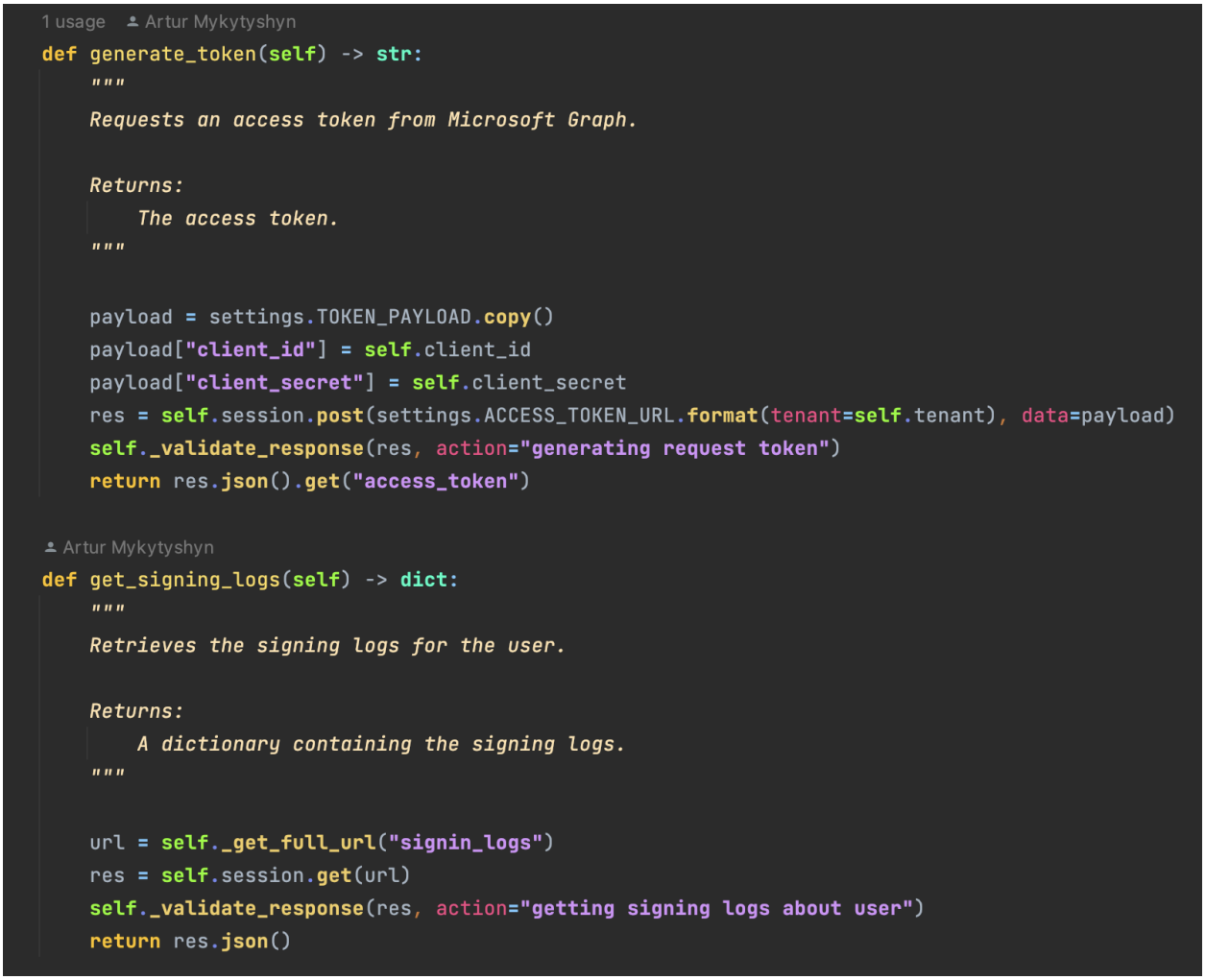


Рисунок 3.4 – Методи генерування токену для авторизації та метод отримання sign-in логів

Основна фаза виконання скрипта зосереджена на зборі даних з Azure Graph API про користувачів і групи та доповненні їх даними з Rusthound. Такий цілеспрямований підхід забезпечує ефективність скрипту у виконанні його основних завдань. Інноваційним кроком є те, що скрипт розширює можливості збору даних за рахунок взаємодії з Rusthound.

Це додаткове джерело даних використовується для збагачення набору даних контекстною інформацією про користувачів і групи. Такі методи, як parse_rusthound_user_data і parse_rusthound_group_data, призначені для отримання та аналізу інформації з даних Rusthound, що сприяє більш повному розумінню організаційного ландшафту.

Метод parse_rusthound_user_data (рисунок 3.5) слугує шлюзом до скарбниці контекстних деталей про окремих користувачів в організації. Взаємодіючи з даними Rusthound, цей метод отримує детальну інформацію, яка виходить за межі Azure Graph API. Це включає, але не обмежується наступним:

```
def parse_rusthound_user_data(self, object_identifier):

    """
    Retrieves information about a user from Rusthound data.

    Args:
        object_identifier (str): The unique identifier of the user.

    Returns:
        dict: A dictionary containing information about the user.
    """
    user_instance = next(filter(lambda u: u["ObjectIdentifier"] == object_identifier, self.rusthound_users), {})

    if user_instance:
        try:

            # Extract relevant information and format it in a dictionary
            user = {
                "SID": user_instance.get("ObjectIdentifier"),
                "domain": user_instance.get("Properties", {}).get("domain"),
                "DN": user_instance.get("Properties", {}).get("distinguishedname"),
                "created": user_instance.get("Properties", {}).get("whencreated"),
                "enabled": user_instance.get("Properties", {}).get("enabled"),
                "passwordchanged": user_instance.get("Properties", {}).get("pwdlastset"),
                "displayname": user_instance.get("Properties", {}).get("displayname"),
                "email": self.make_lowercase(user_instance.get("Properties", {}), "email"),
                "samaccountname": self.make_lowercase(user_instance.get("Properties", {}), "samaccountname"),
                "name": self.make_lowercase(user_instance.get("Properties", {}), "name"),
                "rusthound_groups": [
                    {
                        "name": group.get("RightName"),
                        "SID": group.get("PrincipalSID")
                    } for group in user_instance.get("Aces", []) if group.get("PrincipalType") == 'Group'
                ],
                "rusthound": True
            }
            return user
        except Exception as error:
            self.debug_logger.error(
                "Failed to parse user rusthound data",
                error_message=str(error),
                related_user=user_instance.get("ObjectIdentifier")
            )

    return {"rusthound": False}
```

Рисунок 3.5 – Програма методу аналізу даних від моделі user

Поведінкові шаблони: Rusthound може фіксувати поведінкові шаблони, такі

як тенденції активності користувачів, налаштування доступу та частоту взаємодії. Ці дані дають детальне уявлення про те, як користувачі орієнтуються і використовують ресурси організації.

— Стан безпеки: Rusthound може дати уявлення про стан безпеки окремих користувачів, виявляючи будь-які аномальні моделі поведінки, які можуть вказувати на потенційні ризики для безпеки.

— Міждодаткова активність: Можна виявити взаємодію користувачів у різних додатках і на різних платформах, що проливає світло на їхні взаємопов'язані ролі та обов'язки.

— Історичний контекст: Дані Rusthound можуть надавати історичний контекст, що дозволяє зрозуміти діяльність користувачів та їх еволюцію в часі.

Доповнюючи збагачення даних про окремих користувачів, метод parse_rusthound_group_data заглиблюється в групову динаміку в організації. Він виходить за межі структурної інформації, що надається API Azure Graph, пропонуючи більш цілісне уявлення про те, як групи функціонують і взаємодіють. Основні вдосконалення включають

Метод parse_rusthound_group_data доповнює збагачення даних про окремих користувачів, заглиблюючись у групову динаміку в організації. Він виходить за межі структурної інформації, що надається API Azure Graph, пропонуючи більш цілісне уявлення про те, як групи функціонують і взаємодіють. Основні вдосконалення включають:

— Шаблони співпраці: Дані Rusthound можуть розкривати шаблони співпраці в групах, підкреслюючи, як члени групи співпрацюють над проектами, обмінюються інформацією та роблять внесок у досягнення колективних цілей.

— Ієрархії доступу: Метод може дати уявлення про ієрархію доступу в групах, з'ясовуючи рівні дозволів і привілеїв, призначених різним членам.

— Динамічні зміни в групах: Rusthound може фіксувати динамічні зміни в групах, такі як додавання або видалення членів, що полегшує розуміння еволюції організаційних структур.

— Взаємопов'язані групові відносини: Аналізуючи дані Rusthound, скрипт може розплутати складні взаємозв'язки між різними групами, демонструючи зв'язки співпраці, які можуть бути неочевидними з одного лише API Azure Graph.

Використовуючи дані Rusthound, скрипт дозволяє організаціям приймати більш обґрунтовані рішення на основі детального та цілісного розуміння своїх користувачів і груп. Збагачений набір даних не тільки підсилює зусилля з кібербезпеки, надаючи більш детальну інформацію про стан безпеки, але й підвищує операційну ефективність, виявляючи закономірності, які можуть впливати на розподіл ресурсів, управління доступом і спільні ініціативи.

Незважаючи на додатковий рівень складності, який вносить інтеграція з Rusthound, скрипт підтримує безперебійний робочий процес обробки. Методи аналізу даних Rusthound тісно вплетені в більш широку структуру скрипта, гарантуючи, що збагачені дані безперешкодно інтегруються з інформацією, отриманою з Azure Graph API. Ця єдність має важливе значення для підтримки ефективності та зручності використання сценарію.

По суті, інноваційна інтеграція з Rusthound перетворює скрипт зі звичайного інструменту пошуку даних на складний аналітичний інструмент. Він не тільки йде в ногу з динамічним ландшафтом організаційних даних, але й передбачає і приймає виклики завтрашнього дня. Оскільки організації прагнуть до більш детального розуміння своїх цифрових ландшафтів, цей скрипт є свідченням поєднання інновацій та практичності у сфері науки про дані та кібербезпеки.

3.4 Розробка системи сповіщень про інциденти

У сучасному технологічному ландшафті, де системи генерують величезні обсяги даних, забезпечення своєчасного реагування на інциденти має першорядне значення. Інцидент, який визначається як будь-яка подія, що потенційно може порушити нормальну роботу системи або поставити під загрозу цілісність даних, потребує негайної уваги. Для вирішення цієї проблеми було розроблено автоматизовану систему сповіщення про інциденти, яка використовує можливості мови Python.

У складних IT-інфраструктурах системи моніторингу, таких як ElasticSearch, сповіщення відіграють вирішальну роль у відстеженні подій та потенційних проблем. Однак, великий обсяг даних, що генеруються, може ускладнити ручне виявлення інцидентів та оперативне реагування на них. Звідси виникає потреба в автоматизованій системі сповіщення, яка може інтелектуально виявляти інциденти та оперативно сповіщати про них відповідні зацікавлені сторони.

Клас EmailNotifier слугує комунікаційним центром, відповідальним за надсилання повідомлень. Ця інкапсуляція забезпечує чіткий розподіл завдань, роблячи систему модульною та легкою в обслуговуванні. Клас використовує простий протокол передачі пошти (SMTP) для передачі електронних листів, пропонуючи надійний і широко підтримуваний канал зв'язку.

Клас IncidentDetector інкапсулює логіку виявлення інцидентів. У контексті цього прикладу він включає метод elastic_log_observer_triggered, який імітує виявлення інциденту на основі поведінки Elastic Log Observer. Цей клас розроблено так, щоб його можна було розширювати, що дозволяє інтегрувати більш складні механізми виявлення інцидентів.

Клас NotificationSystem виступає в ролі оркестратора, об'єднуючи класи EmailNotifier та IncidentDetector. Він надає метод check_incident_and_notify, який перевіряє, чи стався інцидент, і при необхідності запускає сповіщення на

електронну пошту. Клас NotificationSystem об'єднує в собі функції виявлення інцидентів та сповіщення про них, а клас CheckIncidentDetector об'єднує функції виявлення інцидентів та сповіщення про них.

Метод check_incident_and_notify перевіряє, чи було виявлено інцидент, і якщо так, то запускає EmailNotifier для надсилання повідомлення електронною поштою. Клас EmailNotifier обробляє передачу електронного листа, інкапсулюючи необхідні дані, такі як адреса відправника, адреса одержувача, тема і тіло листа. Він використовує протокол SMTP для зв'язку і забезпечує надійну обробку помилок.

На закінчення, автоматизована система сповіщення про інциденти, представлена тут, пропонує масштабоване і модульне рішення для вирішення проблем реагування на інциденти в сучасних IT-середовищах. Завдяки використанню принципів ООП, система досягає високого ступеня інкапсуляції, що робить її пристосованою до різноманітних механізмів виявлення інцидентів та каналів зв'язку. Конструктор цього класу зображений на рисунку 3.6.

```python
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

class EmailNotifier:
    def __init__(self, smtp_server, smtp_port, sender_email, sender_password):
        self.smtp_server = smtp_server
        self.smtp_port = smtp_port
        self.sender_email = sender_email
        self.sender_password = sender_password

    def send_email(self, to_email, subject, body):
        msg = MIMEMultipart()
        msg['From'] = self.sender_email
        msg['To'] = to_email
        msg['Subject'] = subject

        msg.attach(MIMEText(body, 'plain'))

        with smtplib.SMTP(self.smtp_server, self.smtp_port) as server:
            server.starttls()
            server.login(self.sender_email, self.sender_password)
            server.sendmail(self.sender_email, to_email, msg.as_string())

class NotificationSystem:
    def __init__(self, email_notifier, incident_detector):
        self.email_notifier = email_notifier
        self.incident_detector = incident_detector

    def check_incident_and_notify(self, to_email):
        if self.incident_detector.elastic_log_observer_triggered():
            subject = "Incident Detected!"
            body = "An incident has been detected in the Elastic Log Observer. Take appropriate action."
            self.email_notifier.send_email(to_email, subject, body)
```

Рисунок 3.6 – Програма надсилання сповіщень

Параметр email_notifier вказує на невід'ємну частину, що відповідає за сповіщення зацікавлених сторін про виявлені інциденти. Важливо, щоб наданий нотифікатор дотримувався чітко визначеного інтерфейсу, що включає метод нотифікації для полегшення комунікації із зовнішнім світом. Надійність забезпечується процесом валідації, який гарантує, що нотифікатор є об'єктом, який можна викликати з необхідним методом.

Аналогічно, параметр incident_detector представляє механізм, який керує ідентифікацією інцидентів у системі. Тут також очікується об'єкт, який можна викликати, а саме об'єкт, оснащений методом detect_incident. Цей метод стає стрижнем для здатності системи розпізнавати та реагувати на аномалії або наперед визначені умови.

Параметр template_path вносить елемент універсальності, дозволяючи користувачам вказувати шлях до своїх файлів шаблонів. У контексті цієї системи шаблонізатор Jinja займає центральне місце, забезпечуючи потужний і динамічний засіб генерації контенту сповіщень. Вказаний шлях ретельно перевіряється на відповідність очікуваним критеріям, що дозволяє уникнути потенційних помилок під час завантаження шаблону.

У класі ініціалізуються template_loader та template_env, що створює основу для використання шаблонів Jinja. Метод render_template, наріжний камінь цієї системи, динамічно рендерить шаблони на основі заданого імені та контексту. Такий модульний підхід дозволяє створювати різноманітні та адаптовані сповіщення, що відповідають різним сценаріям інцидентів.

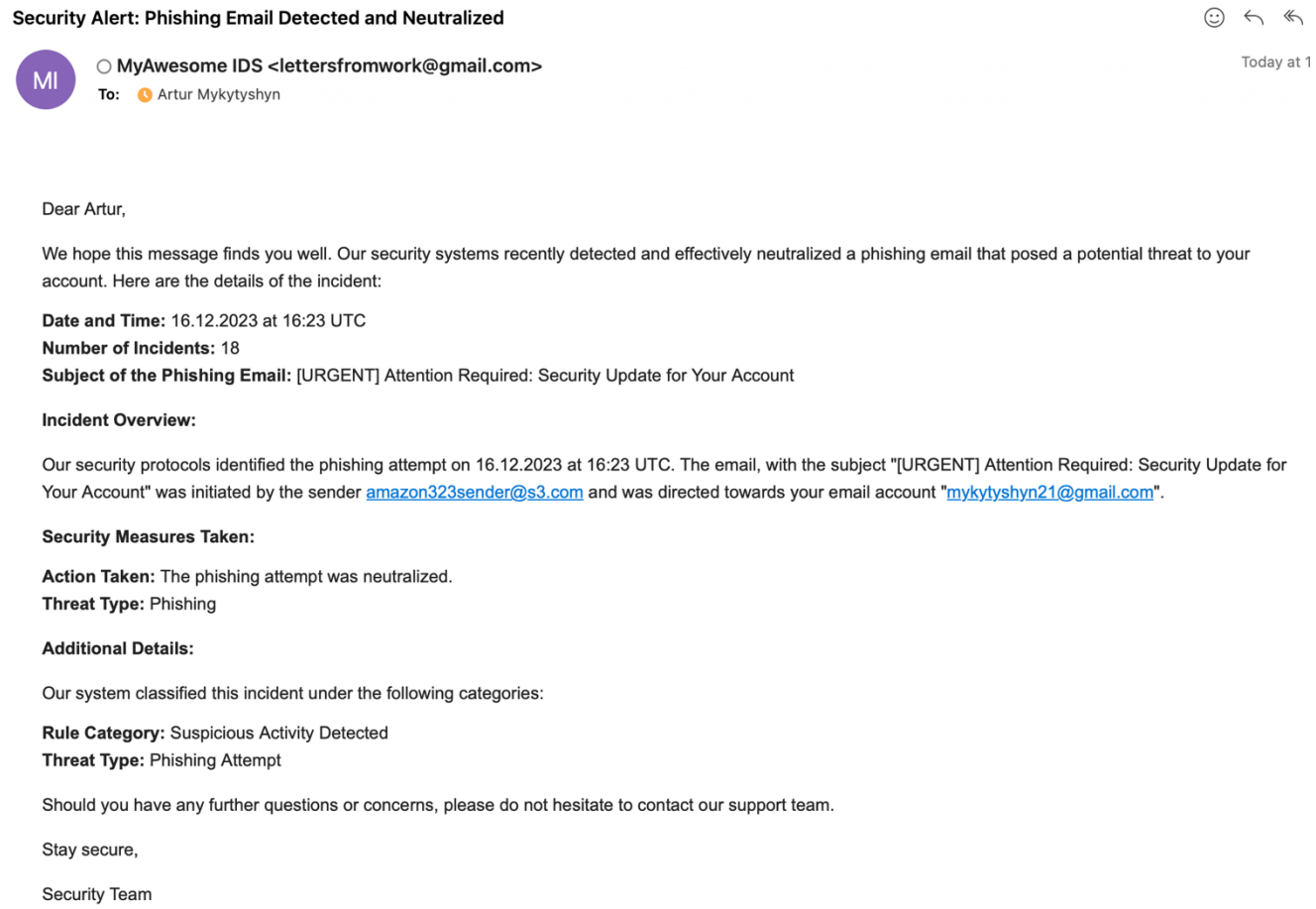
Для утилітарних методів, таких як _validate_notifier, _validate_detector та _validate_path, використовується угода з префіксом підкреслення. Ці методи виконують роль контролерів, захищаючи систему від потенційних неправильних конфігурацій або несумісних компонентів. Застосовуючи ці перевірки, система прагне підтримувати певний рівень стійкості, попереджаючи користувачів про будь-які розбіжності в їхніх конфігураціях на етапі ініціалізації.

По суті, клас NotificationSystem прагне бути чимось більшим, ніж просто оркестром компонентів; він прагне бути всеосяжною, розширюваною і надійною основою для обробки інцидентів і доставки повідомлень. Розумна інтеграція повідомлень електронною поштою, виявлення інцидентів і можливостей шаблонування позиціонує цей клас як універсальний інструмент в арсеналі тих, хто орієнтується в складному ландшафті управління інцидентами. На завершення, розроблений клас NotificationSystem (рисунок 3.7) є ключовим компонентом у комплексній системі управління інцидентами, пропонуючи структуровану та розширювану архітектуру для обробки та сповіщення зацікавлених сторін про виявлені інциденти.

```
import jinja2

class NotificationSystem:
    def __init__(self, email_notifier, incident_detector, template_path):
        self.email_notifier = self._validate_notifier(email_notifier)
        self.incident_detector = self._validate_detector(incident_detector)
        self.template_path = self._validate_path(template_path)
        self.template_loader = jinja2.FileSystemLoader(searchpath=self.template_path)
        self.template_env = jinja2.Environment(loader=self.template_loader)

    def _validate_notifier(self, notifier):
        if not callable(getattr(notifier, "notify", None)):
            raise ValueError("Invalid email notifier provided")
        return notifier

    def _validate_detector(self, detector):
        if not callable(getattr(detector, "detect_incident", None)):
            raise ValueError("Invalid incident detector provided")
        return detector

    def _validate_path(self, path):
        if not isinstance(path, str) or not path.strip():
            raise ValueError("Invalid template path provided")
        return path

    def render_template(self, template_name, context):
        template = self.template_env.get_template(template_name)
        return template.render(context)
```

Рисунок 3.7 – Програма генерування тексту в листі

Злагоджена взаємодія між системою сповіщення електронною поштою, детектором інцидентів та універсальним шаблонізатором Jinja підкреслює адаптивність та відмовостійкість системи.



Security Alert: Phishing Email Detected and Neutralized

MI — MyAwesome IDS <lettersfromwork@gmail.com> — Today at 1

To: Artur Mykytyshyn

Dear Artur,

We hope this message finds you well. Our security systems recently detected and effectively neutralized a phishing email that posed a potential threat to your account. Here are the details of the incident:

Date and Time: 16.12.2023 at 16:23 UTC
Number of Incidents: 18
Subject of the Phishing Email: [URGENT] Attention Required: Security Update for Your Account

Incident Overview:

Our security protocols identified the phishing attempt on 16.12.2023 at 16:23 UTC. The email, with the subject "[URGENT] Attention Required: Security Update for Your Account" was initiated by the sender amazon323sender@s3.com and was directed towards your email account "mykytyshyn21@gmail.com".

Security Measures Taken:

Action Taken: The phishing attempt was neutralized.
Threat Type: Phishing

Additional Details:

Our system classified this incident under the following categories:

Rule Category: Suspicious Activity Detected
Threat Type: Phishing Attempt

Should you have any further questions or concerns, please do not hesitate to contact our support team.

Stay secure,

Security Team

Рисунок 3.8 – Електронне сповіщення про інцидент

На рисунку 3.8 продемонстровано приклад сповіщення про реальний інцидент, яке є автоматично згенерований та надісланий програмою, оскільки серед системних журналів був знайдений саме цей. Лист починається із запевнення, що все гаразд, і сподівання, що все добре. Далі йдеться про те, що системи безпеки нещодавно виявили та ефективно нейтралізували фішинговий лист, який становив потенційну загрозу для вашого акаунта. Деталі інциденту викладені акуратно, включаючи дату і час виявлення, кількість інцидентів (у цьому випадку - 18) і короткий огляд теми фішингового листа.

Мені повідомляється, що спроба фішингу з темою “[URGENT] Attention Required: Security Update for Your Account” була ініційована відправником amazon323sender@s3.com і спрямована на мою поштову скриньку "mykytyshyn21@gmail.com".

У листі не просто повідомляється про це, але й пояснюються вжиті заходи безпеки - спроба фішингу була оперативно нейтралізована. Тип загрози визначено як фішинг, а додаткові деталі вказують на класифікацію інциденту за такими категоріями правил, як "Виявлено підозрілу активність" та "Спроба фішингу". Лист закінчується на дружній ноті, запевняючи, що якщо виникнуть будь-які питання або проблеми, команда підтримки готова допомогти.

Корисність цієї системи сповіщень особливо помітна в контексті складної методології виявлення порушень. Процес починається з отримання журналів від Defender через Filebeat, що забезпечує безперервний потік даних, пов'язаних з безпекою. Однак, визнаючи, що для цілісного розуміння інцидентів безпеки необхідна збагачена контекстна інформація, система легко інтегрується з Rusthound. Rusthound, діючи як допоміжне джерело даних, доповнює необроблені дані журналів детальною інформацією про користувачів і групи.

Цей інноваційний крок у зборі даних додає ще один рівень глибини до процесу виявлення інцидентів. Система використовує спеціальні методи, такі як parse_rusthound_user_data і parse_rusthound_group_data, для отримання і аналізу інформації з Rusthound, що сприяє більш повному розумінню організаційного ландшафту. Таке об'єднання даних з Defender і Rusthound дозволяє системі ідентифікувати і класифікувати порушення безпеки з підвищеною точністю і контекстом.

Організації цього процесу сприяє модульний і добре організований скрипт Python, який демонструє ретельний підхід до взаємодії з Azure Graph API, обробки різних типів даних і взаємодії з базами даних. Включення механізмів обробки

помилок, паралельне виконання для підвищення ефективності та ведення журналів ще більше підвищує надійність та зручність обслуговування скрипту.

Крім того, інтеграція класу NotificationSystem в цей робочий процес забезпечує вирішальний засіб повідомлення про інциденти відповідним зацікавленим сторонам. Завдяки динамічному створенню органів сповіщення за допомогою шаблонів Jinja, спеціальні повідомлення надсилаються електронною поштою, забезпечуючи вичерпне та оперативне інформування одержувачів про виявлені порушення безпеки.

По суті, об'єднання методології виявлення порушень, модульного дизайну скрипту Python та надійних можливостей класу NotificationSystem для сповіщення створює синергетичну екосистему. Ця екосистема не лише виявляє та контекстуалізує інциденти безпеки, але й ефективно повідомляє про них, надаючи організаціям можливість активно реагувати на нові загрози та захищати свої цифрові ландшафти.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Охорона праці

Для ефективної і безпечної роботи колективу працівників з розробки ПЗ комп’ютерних систем, в тому числі і фахівців з підвищення ефективності контролю доступу в приміщення, необхідно організувати безпечні умови праці. При цьому керівник організації несе безпосередню відповідальність за порушення нормативно-правових актів з охорони праці. Окрім цього, на робочих місцях працівників необхідно забезпечити дотримання вимог, затверджених Наказом Мінсоцполітики від 14.02.2018 за № 207 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями». Згідно Вимог приміщення, де розміщені робочі місця операторів, крім приміщень, у яких розміщені робочі місця операторів великих ЕОМ загального призначення (сервер), мають бути оснащені системою автоматичної пожежної сигналізації відповідно до цих вимог:

– переліку однотипних за призначенням об'єктів, які підлягають обладнанню автоматичними установками пожежогасіння та пожежної сигналізації, затвердженого наказом Міністерства України з питань надзвичайних ситуацій та у справах захисту населення від наслідків Чорнобильської катастрофи від 22.08.2005 N 161, зареєстрованого в Міністерстві юстиції України 05.09.2005 за N 990/11270 (НАПБ Б.06.004-2005);

– Державних будівельних норм "Інженерне обладнання будинків і споруд. Пожежна автоматика будинків і споруд", затверджених наказом Держбуду України від 28.10.98 N 247 (далі - ДБН В.2.5-56:2014, з димовими пожежними сповіщувачами та переносними вуглекислотними вогнегасниками.

В інших приміщеннях допускається встановлювати теплові пожежні сповіщувачі. Приміщення, де розміщені робочі місця операторів, мають бути

оснащені вогнегасниками, кількість яких визначається згідно з вимогами ДСТУ 4297:2004 «Пожежна техніка. Технічне обслуговування вогнегасників». Загальні технічні вимоги і з урахуванням граничнодопустимих концентрацій вогнегасної рідини відповідно до вимог НАПБ А.01.001-2014. Приміщення, в яких розміщуються робочі місця операторів сервера загального призначення, обладнуються системою автоматичної пожежної сигналізації та засобами пожежогасіння відповідно до вимог ДБН В.2.5-56:2014, ДБН В.2.5-56:2010, НАПБ А.01.001-2014 і вимог нормативно-технічної та експлуатаційної документації виробника. Проходи до засобів пожежогасіння мають бути вільними.

Лінія електромережі для живлення комп’ютера та периферійних пристроїв повинні бути виконаними як окрема групова трипровідна мережа шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Не допускається використовувати нульовий робочий провідник як нульовий захисний провідник. Нульовий захисний провідник прокладається від стійки групового розподільного щита, розподільного пункту до розеток електроживлення. Не допускається підключати на щиті до одного контактного затискача нульовий робочий та нульовий захисний провідники.

Площа перерізу нульового робочого та нульового захисного провідника в груповій трипровідній мережі має бути не менше площі перерізу фазового провідника. Усі провідники мають відповідати номінальним параметрам мережі та навантаження, умовам навколишнього середовища, умовам розподілу провідників, температурному режиму та типам апаратури захисту, вимогам НПАОП 40.1-1.01-97. У приміщенні, де одночасно експлуатуються понад п’ять комп’ютерів, на помітному, доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення. Комп’ютери повинні підключатися до електромережі тільки за допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення.

У штепсельних з'єднаннях та електророзетках, крім контактів фазового та нульового робочого провідників, мають бути спеціальні контакти для підключення нульового захисного провідника. Їхня конструкція має бути такою, щоб приєднання нульового захисного провідника відбувалося раніше, ніж приєднання фазового та нульового робочого провідників. Порядок роз'єднання при відключенні має бути зворотним. Не допускається підключати комп'ютери до звичайної двопровідної електромережі, в тому числі – з використанням перехідних пристроїв. Електромережі штепсельних з'єднань та електророзеток для живлення комп'ютерної техніки повинні бути виконаними за магістральною схемою, по 3-6 з'єднань або електророзеток в одному колі. Штепсельні з'єднання та електророзетки для напруги 12 В та 42 В за своєю конструкцією мають відрізнятися від штепсельних з'єднань для напруги 127 В та 220 В. Штепсельні з'єднання та електророзетки, розраховані на напругу 12 В та 42 В, мають візуально (за кольором) відрізнятися від кольору штепсельних з'єднань, розрахованих на напругу 127 В та 220 В.

При підвищенні ефективності контролю доступу в приміщення, де для забезпечення безпеки мешканців, співробітників і збереження майна використовуються ДС, важливим, з точки зору охорони праці, є забезпечення достатньої величини природного та штучного освітлення, які визначені у НПАОП 0.00-7.15-18. Організація робочого місця фахівця із дослідження методів та програмно-апаратних засобів оптимізаційних процесів на основі ГА повинна забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним вимогам ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги». Відстань від екрана до ока фахівців, які працюють за комп'ютером визначається згідно з вимогами ДСанПіН 3.3.2.007-98.

Розміщення принтера або іншого пристрою введення-виведення інформації на робочому місці має забезпечувати добру видимість екрана комп'ютера, зручність ручного керування пристроєм введення-виведення інформації в зоні досяжності

моторного поля згідно з вимогами ДСанПіН 3.3.2.007-98.

Таким чином, у результаті аналізу вимог щодо охорони праці користувачів комп'ютерів, визначено особливості організації робочих місць, вимог з електробезпеки, природного та штучного освітлення для ефективної і безпечної роботи фахівців з дослідження автоматичного керування роботизованими механізами.

4.2. Комп'ютерне забезпечення процесу оцінки радіаційної та хімічної обстановки

Екологічне співтовариство розробило сімейство інструментів комплексної екологічної оцінки. Програмне забезпечення і послуги (ESS), комерційна група IIASA, включаючи AirWare (для повітряних проблеми якості), WaterWare (для якості води), CityWare (якість повітря і води в контексті великих міст) і EIAxpert (для надання допомоги із загальним впливом на навколишнє середовище). Функціональність в цілому схожа на RAISON, хоча з великим акцентом на моделювання і меншим акцентом на керування даними. Знову ж таки, інструменти ESS розроблені як модульні набори інструментів (доступні спеціальні системи для вирішення конкретних завдань). Компоненти включають стандартні імітаційні моделі, включаючи моделі ISC і PBM Агентства з охорони навколишнього середовища США, управління даними, в тому числі ГІС, аналіз даних (наприклад, аналіз часових рядів даних спостережень), візуалізація, а також оптимізація.

Іноді немає готових моделей, придатних для конкретного застосування, але тягар розробки нової програми є надмірним. Розробка моделі оточення може відносно легко реалізувати власні моделі комп'ютерів і не турбуватися про включення процедур для вирішення рівнянь, візуалізації і т. д. Як правило, за допомогою цих інструментів користувач просто повинен вказати свою модель, використовуючи або математичні рівняння, або спеціальні графічні символи або

значки, які безпосередньо представляють поведінку системи.

На даний момент є розроблені моделі комп’ютерного забезпечення процесу для оцінки радіаційної та хімічної обстановки.

GEMS – це система на основі моделей, яка підтримує оцінки схильності і ризику, надаючи доступ до одиночних і мультимедійних моделям експозиції, фізико-хімічні властивості методи оцінки, статистичний аналіз, графічні та картографічні програми з відповідними даними на навколишнє середовище, джерела, рецептори і популяції.

HSPF – це комплексний пакет для моделювання кількості і якості стоків з багатоцільових водозборів і процесів радіації, що відбуваються в потоках або повністю змішаних озерах. Це дозволяє інтегроване моделювання землі і грунту, процесів забруднення при гідравлічній і осадово-хімічній взаємодії. Результатом моделювання є тимчасові дані витрати стоку, концентрація поживних речовин і пестицидів, а також дані кількості і якості води в будь-якій точці водозбору. Алгоритми якості води включають динаміку BOD/DO, вуглець, азот і фосфор. Процеси трансформації, які включені в модель це: гідроліз, фотоліз, окислення, випаровування, сорбція і біодеградація. Вторинні або «дочірні» хімічні речовини також моделюються.

Вимоги до даних для моделі можуть бути досить широкими в залежності від конкретного застосування.

Модель MMSOILS – це методологія оцінки впливу на людину і ризику для здоров'я, пов'язаних з викидами забруднень з небезпечних відходів. Мультимедійна модель, що стосується перенесення хімічної речовини в ґрунтові води, поверхневі води, атмосферу і накопичення в їжі. Шляхи впливу на людину, які розглянуті в методології включають: потрапляння в грунт, вдихання летких речовин в повітря і тверді частинки, шкірний контакт, прийом питної води і т.д. Ризик, пов'язаний із загальною дозою опромінення, розраховується на основі хімічної токсичності.

ВИСНОВКИ

Робота над цим проектом розпочалася з ретельного вивчення багатогранного ландшафту захисту інформації. Цей фундаментальний етап включав аналітичний огляд, який розкривав тонкощі різних програмних засобів, що є невід'ємною частиною захисту інформаційних систем.

Потім робота плавно перейшла до більш детального вивчення конкретних захисних заходів. Було ретельно проаналізовано застосування, переваги та особливості брандмауерів, проливаючи світло на їхню ключову роль. Аналогічним чином, детальне обговорення антивірусів і технологій, що лежать в основі віртуальних приватних мереж, забезпечило всебічне розуміння сучасних захисних механізмів.

Потім увага була перенесена на критично важливу сферу систем виявлення вторгнень (IDS). Детальне вивчення архітектури IDS дало цінну інформацію про внутрішню роботу цих систем. У поєднанні з вивченням систем управління інформаційною безпекою ця робота заклала міцний фундамент для розуміння того, як організації керують і реагують на потенційні загрози.

Теоретичне підґрунтя роботи виявилося наріжним каменем. Аналіз системних журналів і тонкощі налаштування передачі цих журналів були ретельно деталізовані. Розробка спеціальної системи виявлення вторгнень була ключовим аспектом, який заклав основу для надійної системи виявлення загроз. Крім того, включення виявлення загроз на основі спостерігачів і систем оповіщення для реагування на інциденти збагатило теоретичну базу.

Перетворивши теорію на практику, робота розпочала шлях практичного розвитку. Створення конвеєра Logstash для аналізу системних журналів стало практичним кроком до практичної реалізації теоретичних висновків. Розробка модулів-спостерігачів для виявлення аномалій ще більше продемонструвала прихильність до активного виявлення загроз. Інтеграція RustHound для покращення

аналізу та розробка системи сповіщення про інциденти додали до теоретичних побудов відчутних шарів.

Отримані результати є не просто кульмінацією серії аналізів і розробок; вони мають глибоку наукову новизну і практичне значення. Робота зробила внесок у розповідь про стійкість до кібербезпеки, наголошуючи на вдосконалених протоколах, активному виявленні загроз та операційній ефективності завдяки інтеграції з RustHound. Адаптована система сповіщення про інциденти стала свідченням практичного застосування теоретичних засад.

Відмінною рисою роботи є її перспективний характер. Замість того, щоб представляти статичні рішення, запропоновані концепції демонструють адаптивність до постійно мінливого ландшафту загроз. Такий далекоглядний підхід позиціонує роботу як динамічну і еволюціонуючу силу в сфері кібербезпеки.

Отже, ця робота є не просто збіркою ідей і розробок, а всебічним дослідженням. Вона охоплює теоретичні висновки, практичну реалізацію і прихильність до міркувань безпеки. Ця робота є цінним ресурсом для організацій, які прагнуть посилити свою кібербезпеку перед обличчям динамічного і складного цифрового ландшафту.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Network Intrusion. [Електронний ресурс]. https://awakesecurity.com/glossary/network-intrusion/ Дата доступу: 07.10.21
2. What is VPN? How It Works, Types of VPN [Електронний ресурс]. https://www.kaspersky.com/resource-center/definitions/what-is-a-vpn Дата доступу: 27.10.21
3. Karen Scarfone, Peter Mell. Guide to Intrusion Detection and Prevention Systems (IDPS). –NIST, - 127 p. https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-94.pdf Дата доступу: 15.11.21
4. Rebecca Bace and Peter Mell. Intrusion Detection Systems. [Електронний ресурс]. URL:https://people.eecs.ku.edu/~hossein/710/Lectures/Readings/08-nist-intrusion-detection.pdf Дата доступу: 25.11.21
5. Микитишин А.Г., Митник М.М., Стухляк П.Д.. Комплексна безпека інформаційних мережевих систем: навчальний посібник – Тернопіль: Вид-во ТНТУ імені Івана Пулюя, 2016. – 256 с.
6. Costin A. Security of cctv and video surveillance systems: Threats, vulnerabilities, attacks, and mitigations. pp. 45–54. [Електронний ресурс] Режим доступа: https://dl.acm.org/doi/abs/10.1145/2995289.2995290 (дата звернення 08.11.2023).
7. Kang M. *Prognostics and Health Management of Electronics: Fundamentals, Machine Learning, and the Internet of Things*. [Електронний ресурс] Режим доступа: https://scholar.google.com/scholar_lookup?title=Prognostics+and+Health+Management+of+Electronics:+Fundamentals,+Machine+Learning,+and+the+Internet+of+Things&author=M.+Kang&publication_year=2018& (дата звернення 08.11.2023).

8. Xu W. Toward human-centered AI: A perspective from human-computer interaction. [Електронний ресурс] Режим доступа: https://dl.acm.org/doi/fullHtml/10.1145/3328485 (дата звернення 08.11.2023).
9. Vojković G., Milenković M., Katulić T. IoT and Smart Home Data Breach Risks from the Perspective of Data Protection and Information Security. [Електронний ресурс] Режим доступа: https://hrcak.srce.hr/ojs/index.php/bsr/article/view/12916 (дата звернення 08.11.2023).
10. Dasgupta A., Gill A.Q., Hussain F. *Internet of Things (IoT) for Automated and Smart Applications*. p. 9. [Електронний ресурс] Режим доступа: https://hrcak.srce.hr/ojs/index.php/bsr/article/view/12916 (дата звернення 08.11.2023).
11. Boerman S.C., Kruikemeier S., Zuiderveen Borgesius F.J. Exploring motivations for online privacy protection behavior: Insights from panel data. [Електронний ресурс] Режим доступа: https://journals.sagepub.com/doi/full/10.1177/0093650218800915 (дата звернення 15.11.2023).
12. van der Sloot B. Where is the harm in a privacy violation. [Електронний ресурс] Режим доступа: https://heinonline.org/HOL/LandingPage?handle=hein.journals/jipitec8&div=38&id=&page= (дата звернення 15.11.2023).
13. Fafoutis X., Marchegiani L., Papadopoulos G.Z., Piechocki R., Tryfonas T., Oikonomou G. Privacy leakage of physical activity levels in wireless embedded wearable systems. [Електронний ресурс] Режим доступа: https://ieeexplore.ieee.org/abstract/document/7792352 (дата звернення 15.11.2023).
14. Davis B.D., Mason J.C., Anwar M. Vulnerability studies and security postures of IoT devices: A smart home case study. [Електронний ресурс] Режим доступа:

https://ieeexplore.ieee.org/abstract/document/9050664 (дата звернення 15.11.2023).

15. Shouran Z., Ashari A., Priyambodo T. Internet of things (IoT) of smart home: Privacy and security. [Електронний ресурс] Режим доступа: https://www.researchgate.net/profile/Zaied-Shouran/publication/331133954_Internet_of_Things_IoT_of_Smart_Home_Privacy_and_Security/links/5c692af14585156b57016c66/Internet-of-Things-IoT-of-Smart-Home-Privacy-and-Security.pdf (дата звернення 15.11.2023).
16. Mai K. *Introduction to Hardware Security and Trust*. p. 175–194. [Електронний ресурс] Режим доступа: https://books.google.com.ua/books?hl=uk&lr=&id=bNiw9448FeIC&oi=fnd&pg=PR3&ots=r37zPcYVow&sig=z7znxSGLSEzews0frkDGPe29JFQ&redir_esc=y#v=onepage&q&f=false (дата звернення 15.11.2023).
17. Conti M., Nati M., Rotundo E., Spolaor R. Mind the plug! laptop-user recognition through power consumption. pp. 37–44. [Електронний ресурс] Режим доступа: https://dl.acm.org/doi/abs/10.1145/2899007.2899009 (дата звернення 15.11.2023).
18. Kocher P., Jaffe J., Jun B., Rohatgi P. *Introduction to differential power analysis*. [Електронний ресурс] Режим доступа: https://link.springer.com/article/10.1007/s13389-011-0006-y (дата звернення 15.11.2023).
19. Kocher P., Jaffe J., Jun B. *Introduction to Differential Power Analysis and Related Attacks*. [Електронний ресурс] Режим доступа: https://www.rambus.com/wp-content/uploads/2015/08/DPATechInfo.pdf (дата звернення 15.11.2023).
20. Li Y., Chen M., Wang J. Introduction to side-channel attacks and fault attacks. pp. 573–575. [Електронний ресурс] Режим доступа:

https://ieeexplore.ieee.org/abstract/document/7522801 (дата звернення 15.11.2023).

21. Deepa G., SriTeja G., Venkateswarlu S. An overview of acoustic side-channel attack. [Електронний ресурс] Режим доступа: https://web.archive.org/web/20180409221150id_/http://www.ijcscn.com/Documents/Volumes/vol3issue1/ijcscn2013030103.pdf (дата звернення 15.11.2023).
22. Новотарський М.А., Нестеренко Б.Б. Штучні нейронні мережі: обчислення // *Праці Інституту математики НАН України*. – Т50. – Київ: Ін-т математики НАН України, 2004. – 408 с.
23. Машинне навчання простими словами. Частина 2. Навчання з підкріпленням. URL: http://www.mmf.lnu.edu.ua/ar/1743 (дата звертання 01.12.2023).
24. Вступ до машинного навчання. URL: http://specials.kunsht.com.ua/ machinelearning2 (дата звертання 01.12.2022).
25. Backes M., Dürmuth M., Gerling S., Pinkal M., Sporleder C.
26. Acoustic {Side-Channel} Attacks on Printers. [Електронний ресурс] Режим доступа: https://www.usenix.org/legacy/event/sec10/tech/full_papers/Backes.pdf (дата звернення 04.12.2023).
27. Cheng P., Bagci I.E., Roedig U., Yan J. SonarSnoop: Active acoustic side-channel attacks. [Електронний ресурс] Режим доступа: https://link.springer.com/article/10.1007/s10207-019-00449-8 (дата звернення 04.12.2023).
28. Alias Y.F., Isa M.A.M., Hashim H. Timing Attack: An Analysis of Preliminary Data. [Електронний ресурс] Режим доступа: https://jtec.utem.edu.my/jtec/article/view/1774 (дата звернення 04.12.2023).
29. Joshi M., Hadi T.H. A review of network traffic analysis and prediction techniques. [Електронний ресурс] Режим доступа: https://arxiv.org/abs/1507.05722 (дата звернення 04.12.2023).

30. Srinivasan V., Stankovic J., Whitehouse K. Protecting your daily in-home activity information from a wireless snooping attack. pp. 202–211. [Електронний ресурс] Режим доступа: https://dl.acm.org/doi/abs/10.1145/1409635.1409663 (дата звернення 04.12.2023).
31. Noto M., Sato H. A method for the shortest path search by extended Dijkstra algorithm. pp. 2316–2320. [Електронний ресурс] Режим доступа: https://ieeexplore.ieee.org/abstract/document/886462 (дата звернення 04.12.2023).
32. Навчальний посібник «ТЕХНОЕКОЛОГІЯ ТА ЦИВІЛЬНА БЕЗПЕКА. ЧАСТИНА «ЦИВІЛЬНА БЕЗПЕКА»» / автор-укладач В.С. Стручок– Тернопіль: ФОП Паляниця В. А., – 156 с.
33. Методичний посібник для здобувачів освітнього ступеня «магістр» всіх спеціальностей денної та заочної (дистанційної) форм навчання «БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ» / В.С. Стручок –Тернопіль: ФОП Паляниця В. А., –156 с.
34. Вовк Ю. Я. Охорона праці в галузі. Навчальний посібник / Ю. Я. Вовк, І. П. Вовк – Тернопіль: ФОП Паляниця В.А. – 2015. – 172 с.

ДОДАТОК А - Тези конференції

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

XI НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

13-14 грудня 2023 року

ТЕРНОПІЛЬ
2023

УДК 004.056.5
А.А. Микитишин, Т.А. Лечаченко, д-р. філософії, ст.викл.
Тернопільський національний технічний університет імені Івана Пулюя, Україна

АНАЛІЗ МЕТОДИК ВИЯВЛЕННЯ ВТОРГНЕНЬ У СИСТЕМАХ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ

A. A. Mykytyshyn, T. A. Lechachenko, Ph.D., Senior Lecturer

ANALYSIS OF INTRUSION DETECTION METHODS IN INFORMATION SECURITY SYSTEMS

На сьогоднішній день захист інформації є дуже важливою складовою нашого життя. Інформаційні системи стрімко розвиваються і разом з цим зростає кількість загроз та вразливостей до яких ці системи чутливі. Тому механізми та комплекси виявлення та запобігання вторгнень є дуже важливим напрямком в сфері комп'ютерної безпеки. Не можна недооцінювати ризики пов'язані з порушенням конфіденційності, цілісності та доступності даних.

Вторгнення в мережу – це будь-яка несанкціонована діяльність в цифровій мережі. Вторгнення в мережу часто пов'язані з крадіжкою цінних мережевих ресурсів і майже завжди ставлять під загрозу безпеку мереж або їх даних [1].

Виявлення кібератак має важливе значення для забезпечення безпеки мереж і систем. Тому, щоб виявити зловмисну діяльність, багато компаній використовують платформи які забезпечують захист інформації та здійснюють керування подіями, щоб збирати, керувати та аналізувати дані з різних джерел журналів, таких як брандмауери та операційні системи.

Загалом, зловмисну активність можна виявити, реалізувавши правила виявлення в таких системах, які перевіряють шаблони активності або код, що відповідають атакам, тобто сигнатури зловмисної діяльності. Однак, незважаючи на те, що виявлення атак за допомогою сигнатур добре працює в багатьох ситуаціях, це також створює деякі проблеми та обмеження, такі як труднощі у виявленні варіантів атак або невідомих атак, оскільки для них немає спеціальних сигнатур [2].

Щоб подолати ці обмеження, все частіше застосовують інший підхід до виявлення: виявлення аномалій. Системи виявлення аномалій моделюють нормальну поведінку користувачів, мереж і систем, щоб встановити нормальні шаблони та виміряти відхилення від цих параметрів, оскільки значні відхилення представляють невідповідності з нормальною поведінкою і, отже, вказують на підозрілу активність.

В результаті дослідження, на основі архітектури ElasticSearch, розроблено та впроваджено алгоритм машинного навчання, спрямований на виявлення аномалій у системних та мережевих даних. Цей алгоритм базується на вивченні нормальних шаблонів та виявленні відхилень, що вказують на можливі загрози. Програмне забезпечення на Python буде використано для аналізу та інтеграції цих подій у систему виявлення аномалій на базі ElasticSearch. Цей комплексний підхід дозволить забезпечити ефективне виявлення та реагування на потенційні кіберзагрози, забезпечуючи надійний рівень безпеки інформаційних систем

Література

1. Network Intrusion. [Електронний ресурс]. URL:https://awakesecurity.com /glossary /network-intrusion/ Дата доступу: 07.10.21

2. Liao, Hung-Jen, et al. "Intrusion detection system: A comprehensive review." Journal of Network and Computer Applications 36.1 (2013): 16-24.