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

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня
бакалавр
(назва освітнього ступеня)

на тему: Автоматизація моніторингу витоку інформації

Виконав: студент (ка) IV курсу, групи СБс-42
спеціальності 125 Кібербезпека
(шифр і назва спеціальності)

		Покидко О.В.
	(підпис)	(прізвище та ініціали)
Керівник		Кареліна О.В.
	(підпис)	(прізвище та ініціали)
Нормоконтроль		Кареліна О.В.
	(підпис)	(прізвище та ініціали)
Завідувач кафедри		Загородна Н.В.
	(підпис)	(прізвище та ініціали)
Рецензент		
	(підпис)	(прізвище та ініціали)

м. Тернопіль – 2021

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
Кафедра кібербезпеки

ЗАТВЕРДЖУЮ
Завідувач кафедри Загородна Н. В.
(підпис)
«_____» _____ 202__ р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)
за спеціальністю 125 Кібербезпека
(шифр і назва спеціальності)
Студенту Покидку Олександру Вікторовичу
(прізвище, ім'я, по батькові)
1. Тема роботи Автоматизація моніторингу витоку інформації
Керівник роботи Кареліна Олена Володимирівна, кандидат педагогічних наук, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом по університету від «_____» _____ 202__ року №_____
2. Термін подання студентом роботи
3. Вихідні дані до роботи
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз поняття витоку інформації. 1.1. Проблема витоку інформації.
1.2. Корпоративні загрози витоку інформації. 1.3. Сучасні рішення попередження витоку.
Інформації. 1.4. Аналіз готових рішень поставленого завдання. 2. Опис функціоналу
Системи виявлення витоків даних. 2.1. Характеристика системи виявлення витоку
Даних. 2.2. Математичні методи. 3. Програмна реалізація засобу автоматичного моніторингу
витоку інформації. 3.1. Проектування відношень компонентів автоматизованої системи.
3.2. Програмна реалізація модуля пошуку та збору інформації з дарк вебу.
3.3. Програмна реалізація модулю аналізатора файлів. 3.4. Тестування функціональних
можливостей. 4. Безпека життєдіяльності, основи охорони праці. 4.1. Долікарська
допомога при ураженні електричним струмом. 4.2. Вимоги до виробничого
освітлення та його нормування. 4.3. Вимоги пожежної безпеки при гасінні електроустановок.
Висновки. Список використаних джерел.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Гурик О. Я. к.т.н., доцент		

7. Дата видачі завдання

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка

Студент _____ Покидко О. В.
(підпис) (прізвище та ініціали)

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

АНОТАЦІЯ

Автоматизація моніторингу витоку інформації. // Кваліфікаційна робота освітнього рівня «Бакалавр» // Покидко Олександр Вікторович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБс-42 // Тернопіль, 2021 // С. 69, рис. – 20 , табл. – 0, кресл. – 0, додат. – 3, бібліогр. – 31.

Ключові слова: Python, витік даних, порушення даних, автоматизація, аналіз, загроза, пошук ключових слів.

У кваліфікаційній роботі розроблено інструмент для автоматичного моніторингу витоку інформації базуючись на пошуку ключових слів у файлах.

Інструмент для автоматичного моніторингу витоку здатний проводити аналіз широкого спектру текстових файлів та зображень на вміст заданих ключових слів. Базуючись на вмісті ключових слів у файлі, а також наявність у ньому, доменів, електронних адрес та номерів банківських карт система дає передбачення причетності файлу до проблеми витоку даних. Окрім цього, реалізовано інструмент збору файлів з дарк вебу для їх подальшого аналізу.

Програмна реалізація виконана мовою програмування Python, з використанням інструментів LibreOffice, Google Tesseract OCR, Tor, а також бібліотек textract, zipfile, pyexifinfo, pdf2image, Stem.

ANNOTATION

Automation of information leak monitoring // Qualification work of Bachelor educational degree // Pokydko Oleksandr // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Cybersecurity Department, SBs-42 group // Ternopil, 2021 // Pages – 69, figures – 20, tables – 0, sketches - 0, addendums - 3, references - 31.

Keywords: Python, data leak, data breach, automation, analysis, threat, keywords search.

In the qualification work developed a tool for automatic monitoring of information leakage based on the search for keywords in files.

The tool for automatic leakage monitoring is able to analyze a wide range of text files and images for the content of specified keywords. Based on the content of keywords in the file, as well as the presence in it, domains, email addresses and bank card numbers, the system predicts the involvement of the file in the problem of data leakage. In addition, a tool for collecting files from the dark web for their further analysis has been implemented.

The software implementation is performed in the Python programming language, using tools such as: LibreOffice, Google Tesseract OCR, Tor, and also libraries textract, zipfile, pyexifinfo, pdf2image, Stem.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

DLD (Data Leak Detection) - виявлення витоків даних.

ЗВІ - Запобігання витоку інформації.

DLP (Data Leak Prevention) - запобігання витоків даних.

HIPAA (Health Insurance Portability and Accountability Act) - акт про мобільність та підзвітність медичного страхування.

DoS (Denial of Service) - напад на комп'ютерну систему з наміром зробити комп'ютерні ресурси недоступними користувачам, для яких комп'ютерна система була призначена.

IDS (Intrusion Detection System) - програмний або апаратний засіб, призначений для виявлення фактів несанкціонованого доступу в комп'ютерну систему або мережу або несанкціонованого управління ними в основному через Інтернет.

MITM (Man in the middle) - атака «людина посередині» коли атакувальник здатний читати та видозмінювати за своєю волею повідомлення, якими обмінюються кореспонденти.

SQL (Structured query language) ін'єкція - один з поширених способів злому сайтів та програм, що працюють з базами даних, заснований на впровадженні в запит довільного SQL-коду.

SOC (Security operations center) - централізований підрозділ, який займається питаннями безпеки на організаційно-технічному рівні в компанії.

SIEM (Security information and event management) - підрозділ у галузі комп'ютерної безпеки, що забезпечує аналіз у режимі реального часу попереджень про безпеку, що генеруються додатками та мережевим обладнанням.

DLPD (Data leak prevention and detection) - запобігання та виявлення витоків даних.

ВСТУП

Витік даних є серйозною загрозою для діяльності підприємств, таких як корпорації та державні установи. Втрата конфіденційної інформації може призвести до значного збитку для репутації та фінансових втрат і навіть може зашкодити довготривалій стабільності організації. Поширені типи інформації, що просочується, варіюються від даних працівника / замовника, інтелектуальної власності, персональні дані користувачів та до медичних записів. Згідно з дослідженням IBM Cost of Data Breach Data 2020 [1], середня консолідована вартість порушення даних досягла 3,8 мільйонів доларів. Прогноз Juniper Research [2] передбачає, що загальна річна вартість порушень даних досягне глобальних показників до 5 мільярдів доларів до 2024 року. За останні кілька років було багато помітних випадків втрати даних, які коштували компаніям мільйони доларів. Самим недавнім можна назвати взлам Colonial Pipelines [3] за який компанія заплатила 4.4 мільйони доларів.

Оскільки в цифрову еру обсяг даних зростає в геометричній прогресії, а витоки даних трапляються частіше, ніж будь-коли раніше, запобігання витоку конфіденційної інформації до несанкціонованих сторін стає одним з найбільш актуальних проблем безпеки для підприємств.

Метою систем виявлення витоків даних (DLD) є моніторинг, ідентифікація, та сповіщення ненавмисному чи навмисному витоку конфіденційної інформації в середовищі підприємства.

Витік даних може бути спричинений внутрішніми та зовнішніми порушеннями інформації, навмисно (наприклад, викрадення даних зловмисниками або саботаж зловмисниками) або ненавмисно (наприклад, випадкове розголошення конфіденційної інформації працівниками та партнерами). Мотивація інсайдерських атак різноманітна, включаючи шпигунство, незадоволення керівництвом або через фінансову винагороду. Випадкові витоки в основному виникають внаслідок ненавмисної діяльності через погано налаштований бізнес-процес, погано сформовані політики безпеки, чи через некомпетентність працівників.

Актуальність теми кваліфікаційної роботи пов’язана зі значним поширенням явища витоку даних і базується на проблемі, що дуже мала кількість компаній задумуються про необхідність подібного моніторингу. Використання автоматичної системи витоку інформації дозволяє компаніям отримати інформацію про витік даних який уже відбувся, що дає змогу проаналізувати скомпрометовані дані та зреагувати на інцидент.

Мета цієї - висвітлити загрози витоку даних на підприємствах, систематизувати рішення для виявлення та запобігання витоку даних, а також реалізувати автоматичну систему виявлення витоку інформації, яка базується на вмісті документу за допомогою виявлення ключових слів.

Рішення щодо проектування даної автоматичної системи моніторингу витоку інформації було прийняте після розгляду інцидентів витоку інформації, дослідження роботи існуючих рішень, аналізу літературних джерел та математичних моделей.

Результати роботи пройшли апробацію на науково-технічної конференції „Інформаційні моделі, системи та технології“, 9-10 грудня 2020 року. Розроблене програмне рішення для автоматизації моніторингу витоку корпоративної інформації впроваджене у роботу компанії з кібербезпеки Cyberoo.

Пояснювальна записка обсягом 69 сторінок містить, 20 рисунків, 9 формул, 8 лістингів, 3 додатки. Список використаних джерел розміщується на 3 сторінках і містить 31 джерел.

1. АНАЛІЗ ПОНЯТТЯ ВИТОКУ ІНФОРМАЦІЇ

1.1. Проблема витоку інформації

Порушення даних та витік даних – це терміни, на які можна натрапити час від часу. Вони звучать настільки схоже, що їх легко переплутати між собою. Зрештою, обидва означають, що хтось отримав доступ до інформації, яку він не мав права бачити. Розглянемо різницю між цими двома термінами детальніше.

Витік даних (data leak) - це несанкціонована передача даних із організації до зовнішнього пункту призначення або одержувача. Цей термін можна використовувати для опису даних, які передаються в електронному або фізичному плані. Загрози витоку даних зазвичай трапляються через Інтернет та електронну пошту, але також можуть виникати через мобільні пристрої зберігання даних, такі як оптичні носії, USB-ключі та ноутбуки. Можливо, хтось випадково виявив вразливість в сервісі чи додатку, яка дозволила отримати несанкціонований доступ до інформації. Або компанія погано обробляла інформацію, і в підсумку вона була скомпрометована через погану практику безпеки. Витік даних також може бути наслідком помилки, коли наприклад електронне повідомлення надіслане не тому отримувачу. Але врешті-решт дані скомпрометовані.

Порушенню даних (data breach) - це прямий напад на приватні дані несанкціонованим суб’єктом. Існує безліч прикладів порушення даних. Наприклад, хакери, які проникають у комп’ютерну базу даних, або хтось, хто примушує вас надавати доступ до даних, яких вони не повинні мати. Атака здійснюється з наміром викрасти дані, через це постійно трапляються все більше і більше порушень даних, і це прибутковий бізнес для кіберзлочинців. Найбільші порушення можуть включати мільярди записів особистої інформації.

Загроза для кінцевих користувачів та компаній однакова в будь-якому випадку.

Така ж ситуація і з порушеннями даних та витоками даних. Дані були скомпрометовані, незалежно від того, сталося це через порушення чи витік

інформації. Викрадені дані можуть призвести до викрадення персональних акаунтів, втрати корпоративної таємниці і навіть крадіжки особистості [4]. Тому при витоку інформації потрібно переконатись, що яку б інформацію не було вкрадено, вона не може бути використана проти вас. Наприклад, якщо інформація про кредитну картку була скомпрометована, цю картку слід заморозити у то й ж момент.

Якщо в результаті порушення даних або витоку інформації було вкрадено паролі, все одно необхідно діяти, поки не буде пізно. Можливо, той, хто має доступ до даних, ще не використав пароль для управління обліковим записом. Негайна зміна скомпрометованих паролів може врятувати від великих клопотів.

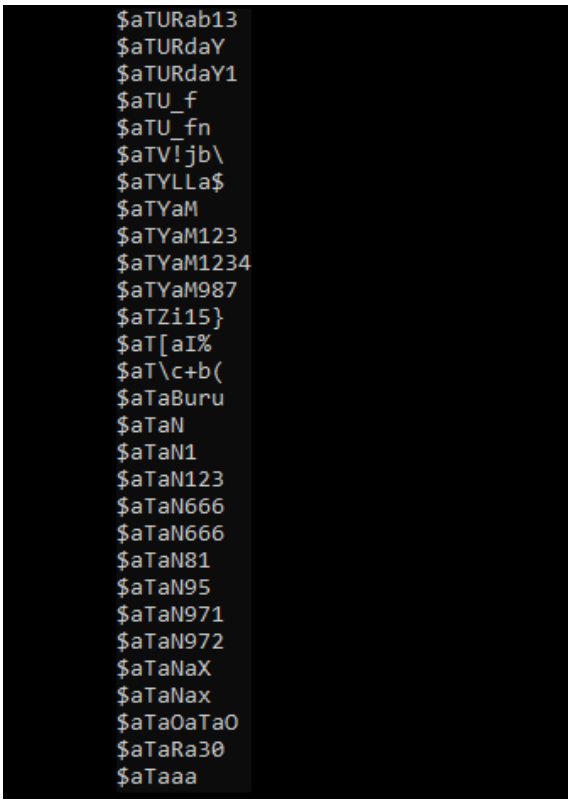

Рисунок 1.1 – Приклад паролів з свіжого data breach який вміщає 8,45 мільярдів паролів [5].

Не проходить і дня без чергової новини про порушення конфіденційності інформації у масових медіа. Витік даних, також відомий як викрадення даних, є величезною проблемою в сфері безпеки даних, і шкода, заподіяна будь-якій організації, незалежно від розміру та галузі, може бути серйозною. Починаючи зі

зменшення доходів до заплямованої репутації або масових фінансових покарань чи дорогих судових процесів [6] [7], це загрози, від яких будь-яка організація захоче захиститися.

Існує багато різних типів витоку даних, і важливо розуміти, що проблема може походити через зовнішні або внутрішні джерела. Захисні заходи повинні охоплювати всі сфери, щоб запобігти найпоширенішим загрозам витоку даних, тому загрози можна поділити на наступні типи:

— Випадковий витік. "Несанкціонований" витік даних не обов'язково означає запланований або зловмисний. Хороша новина полягає в тому, що більшість випадків витоку даних є випадковими. Наприклад, працівник може ненавмисно вибрати неправильного одержувача під час надсилання електронного листа, що містить конфіденційні дані. На жаль, ненавмисний витік даних все одно може спричинити ті самі проблеми, оскільки такі помилки не пом'якшують юридичну відповідальність.

— Незадоволений працівник. [8] Коли ми думаємо про витоки даних, ми думаємо про дані, що зберігаються на викрадених або недоречних ноутбуках, або дані, які просочуються по електронній пошті. Однак переважна більшість втрат даних відбувається не на електронному носії, це відбувається за допомогою принтерів, фотоапаратів, фотографій, знімних USB-накопичувачів та навіть нишпорення у документах викинутих у сміття. Хоча працівник, можливо, підписав трудовий договір, який фактично означає довіру між роботодавцем та працівником, ніщо не заважає їм пізніше поширювати конфіденційну інформацію, якщо вони незадоволені по певних причинах або їм обіцяють значну виплату з боку кіберзлочинців. Цей тип витоку даних часто називають вилученням даних.

— Цільова атака. Багато організацій надають працівникам доступ до Інтернету, електронної пошти та обміну миттєвими повідомленнями як частину своєї роботи. Проблема полягає в тому, що всі ці носії здатні передавати файли або отримувати доступ до зовнішніх джерел через Інтернет. В свою чергу кіберзлочинці користуються такою глобальною підключеністю працівників для отримання неавторизованого доступу до конфіденційних даних. Для цього можуть

використовуватись як і шкідливе програмне забезпечення так і методи соціальної інженерії. Також атака може бути спрямована не тільки на користувача пристрою, а й на вразливе програмне забезпечення, тому витік боже відбутися навіть без активної участі користувача.

До ризику витоку інформації, як і до інших ризиків можна застосувати наступні стратегії:

— Уникнення. Уникнення реалізується в захисті критичних систем зберігання інформації, налаштуванні певних правил доступу до інформації, відслідковування змін та сповіщення та використання шифрування.

— Прийняття. Подібне рішення вважається не припустимим але цілком реальним.

— Моніторинг, підготовка та реагування. Основна ціль даної роботи.

— Пом’якшення. Реалізовується за принципом зберігання найменш важливих даних.

— Перенесення. Всі втрати будуть покриті страховими компаніями, або компаніями на замовлення які реалізовували системи захисту.

Ціллю даної роботи є створення системи автоматичного моніторингу інформації, яка опинилася у несанкціонованих осіб, тому нам не важливо яким чином вона була отримана – через сплановану атаку чи через неправильне керування даними.

1.2. Корпоративні загрози витоку інформації

Витік даних є серйозною загрозою для діяльності підприємств, таких як корпорації та державні установи. Втрата конфіденційної інформації може призвести до значного збитку для репутації та фінансових втрат і навіть може зашкодити довготривалій стабільності організації. Поширені типи інформації, що просочується, варіюються від даних про працівників / клієнтів, інтелектуальної власності до медичних записів. Згідно з дослідженням IBM Cost of Data Breach Data 2016, 1 середня консолідована вартість порушення даних досягла 4 мільйонів доларів.

Оскільки в цифрову еру обсяг даних зростає в геометричній прогресії, а витоки даних трапляються частіше, ніж будь-коли раніше, запобігання витоку конфіденційної інформації до несанкціонованих сторін стає однією з найбільш актуальних проблем безпеки проблеми для підприємств.

Витік даних може бути спричинений внутрішніми та зовнішніми порушеннями інформації або навмисно (наприклад, викрадення даних зловмисниками або саботаж зловмисниками) або ненавмисно (наприклад, випадкове розголошення конфіденційної інформації працівниками та партнерами). Дослідження Intel Security5 показало, що внутрішні співробітники є причиною 43% витоків корпоративних даних, і половина цих витоків є випадковими.

Мотивація інсайдерських атак різноманітна, включаючи шпигунство підприємств, скарги з боку роботодавця або фінансову винагороду. Випадкові витоки в основному виникають внаслідок ненавмисної діяльності через поганий бізнес-процес, такий як незастосування відповідних профілактичних технологій та політики безпеки, або нагляд працівників.

Один із підходів до класифікації загроз витоку даних базується на їх причинах, навмисних або ненавмисних витоків конфіденційної інформації.

Інший підхід базується на тому, які сторони спричинили витік: інсайдери чи аутсайдери. Як показано на рисунку 1.2, навмисні витоки трапляються через зовнішні сторони або зловмисних інсайдерів. Порушення зовнішніх даних, як правило, спричинене зламами хакерів, шкідливим програмним забезпеченням, вірусами та соціальною інженерією. Наприклад, супротивник може використовувати системний бекдор або неправильно налаштований контроль доступу, щоб обійти механізм автентифікації сервера та отримати доступ до конфіденційної інформації.

Атаки соціальної інженерії (наприклад, фішинг) стають дедалі складнішими проти підприємств, обдурюючи працівників та приватних осіб щодо передачі цінних даних про компанію кіберзлочинцям. Внутрішній витік даних може бути спричинений або навмисними діями (наприклад, через шпигунство за фінансову винагороду або скарги працівників) або ненавмисними помилками (наприклад,

випадковий обмін даними працівниками або передача конфіденційних даних без належного шифрування).

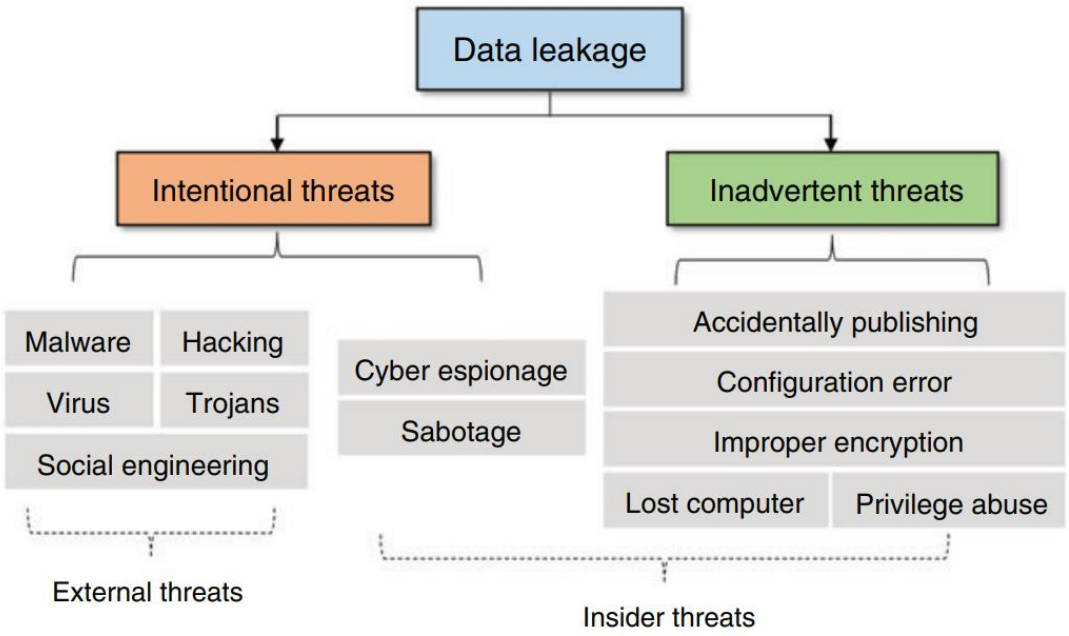

Рисунок 1.2 – Класифікація корпоративних загроз витоку даних [9]

Результати показують, що понад 60% порушень даних були спричинені інсайдерами, це підкреслює, що як технологічні, так і нетехнологічні заходи є важливими для запобігання порушенням даних. Витоки даних також можна охарактеризувати на основі інших ознак, наприклад, за галузевим сектором або за типом подій.

Існує постійний аргумент щодо рейтингу, але п’ятірка галузей, у яких найбільша небезпека порушення даних, включає:

- охорона здоров'я;
- розміщення житла;
- державний сектор;
- роздрібна торгівля;
- фінанси.

Організації охорони здоров’я, як правило, страждають від інсайдерських загроз більше, ніж організації будь-якого іншого сектору. 56% усіх суб'єктів загроз в

організаціях охорони здоров'я походять зсередини, як зазначено у звіті про розслідування порушень даних Verizon 2018. На рисунку 1.3 показано інформацію щодо сектору охорони здоров`я стосовно явища витоку даних (дані надано Verizon 2019 та HIPAA).

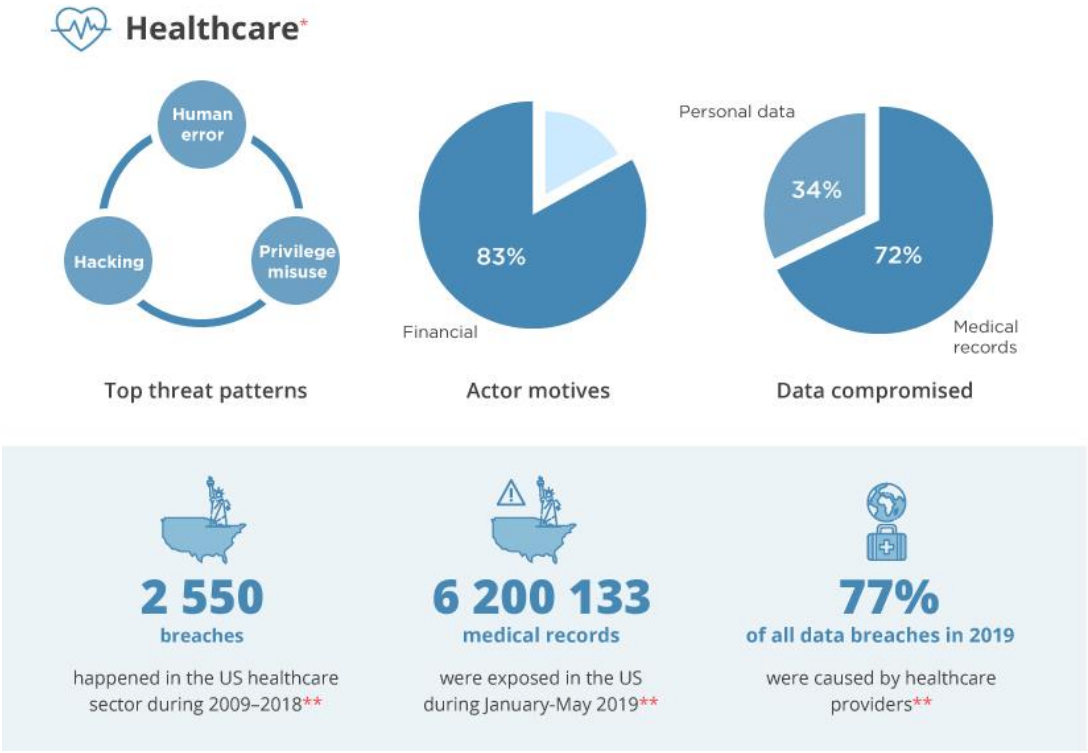

Рисунок 1.3 – Інформація стосовно витоку даних в медичній галузі [10]

Індустрія туризму в останні роки дуже постраждала, проте готелі досі збирають платіжну інформацію та приватні дані від клієнтів. На жаль вони, як правило, вкладають обмежену кількість ресурсів в комп'ютерну безпеку та системи зменшення ризиків на основі користувачів. Конфіденційні дані в кінцевому підсумку є легко доступними для працівників готелю та сторонніх постачальників.

Ще одним тривожним фактом є те, що 96% усіх порушень розміщення житла не виявляються впродовж місяців після інциденту. Як правило, готель дізнається про інцидент через розслідування правоохоронних органів.

Відпустка в готелі, який не звертає уваги на свою кібербезпеку, може мати несподівані наслідки. Наприклад, дані, надані 500 мільйонами гостей готелю Marriott Hotels, були викрадені через погану практику безпеки. Компанія розкрила

це порушення в 2018 році. Хакери отримали несанкціонований доступ до бази даних для гостей, що була забронювана в готелі Marriott, купленої в 2016 році. Бекдор не був виявлений до 2018 року. У витоку інформації містилися імена клієнтів, електронні адреси, номери телефонів, номери паспортів та кредитних карток, дати народження. Рисунок 1.4 демонструє масштаб проблеми витоку інформації в секторі розміщення житла.

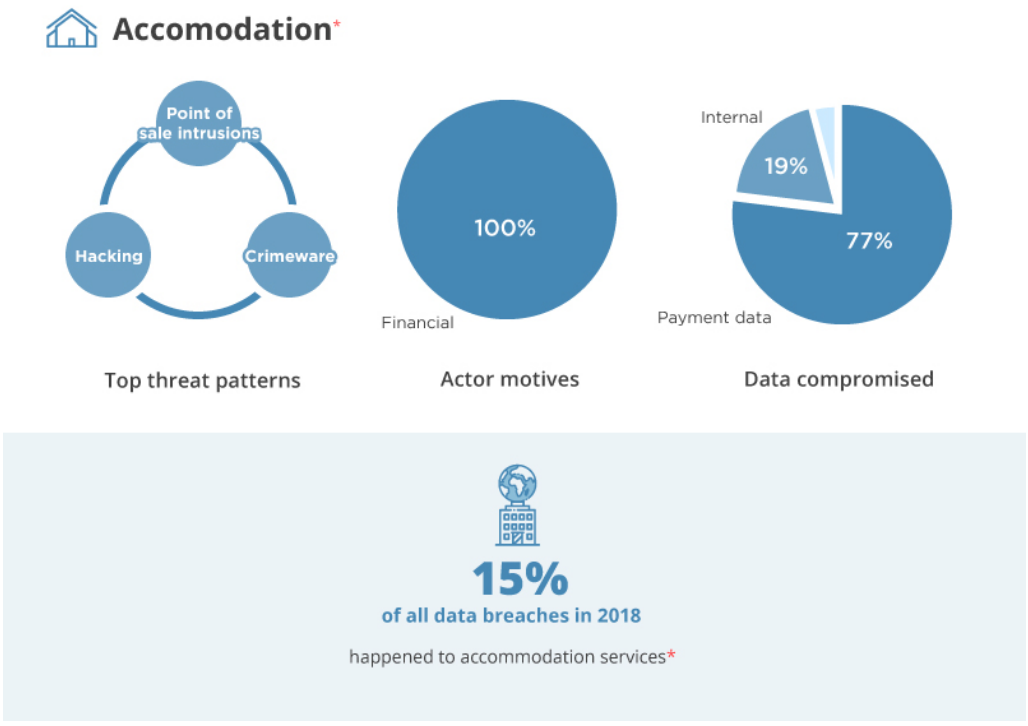

Рисунок 1.4 – Інформація стосовно витоку даних в галузі розміщення житла [10]

Деякі дослідники ставлять державний сектор на перше місце у списку галузей, постраждалих від кібератак. Дані уряду часто є викраденими через шпигунство чи фінансову вигоду.

Ситуація погіршується через брак інвестицій у кібербезпеку. Комплексні системи безпеки та моніторингу не передбачені бюджетом, не мають пріоритетів або гальмують роботу агентства.

На даний момент конфіденційні та секретні записи можуть потрапити в чужі руки. Департамент соціальних служб штату Орегон розкрив масштабне порушення даних у 2019 році. Співробітник DHS відкрив фішинг-посилання та розкрив свої

повноваження кіберзлочинцям. Використовуючи його, хакери отримали електронні адреси та особисту інформацію 645 тисяч людей. Рисунок 1.5 ілюструє статистичні дані стосовно явища витоку даних в державному секторі.

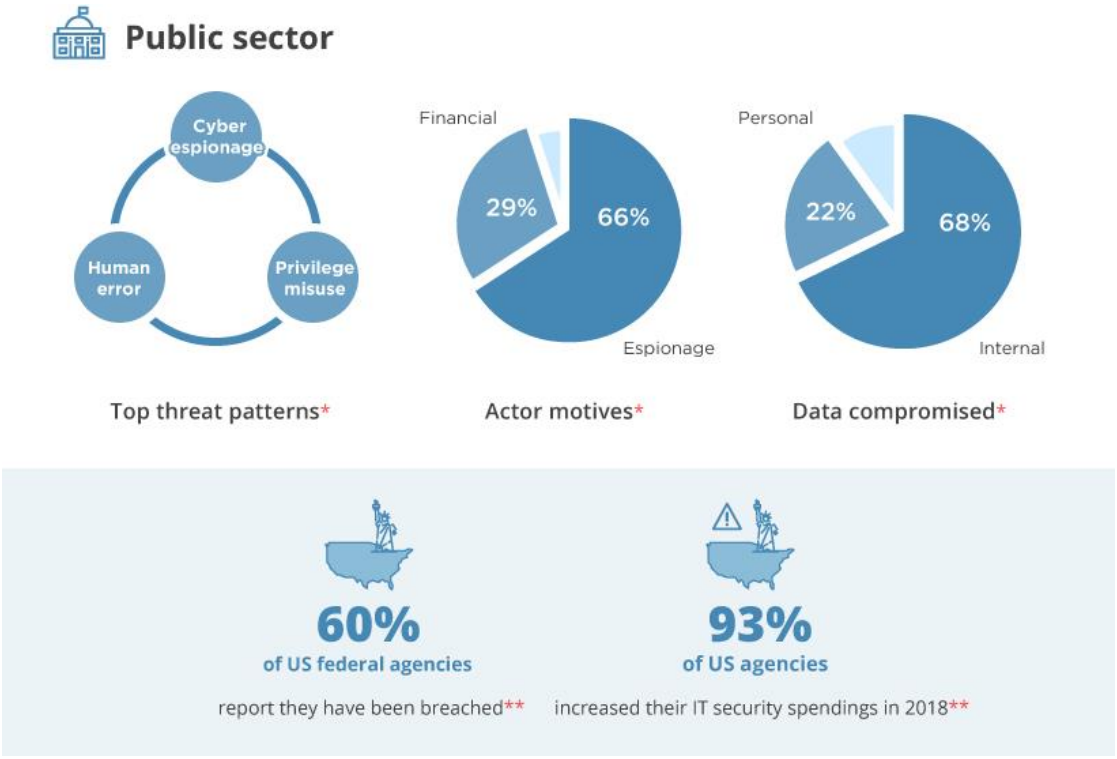

Рисунок 1.5 – Інформація стосовно витоку даних в державному секторі [10]

Торгівля завжди була предметом шахрайства. Наше цифрове століття привносить у цю галузь інструменти цифрового шахрайства. Роздрібні продавці часто страждають від DoS-атак на свої веб-сайти. Роздрібні продавці покладаються на сторонні організації для надання послуг безпеки або взагалі не турбуються про це.

Одне з найвідоміших порушень даних у роздрібній торгівлі сталося через нехтування правилами безпеки. На роздрібну мережу Target було здійснено напад на День Подяки 2013 року – найнапруженіший час для будь-якого роздрібного продавця. Хакери отримали доступ до пристроїв для зчитування платіжних карток сторонніх постачальників, отримавши дані про контактні дані та кредитні картки 110 мільйонів клієнтів.

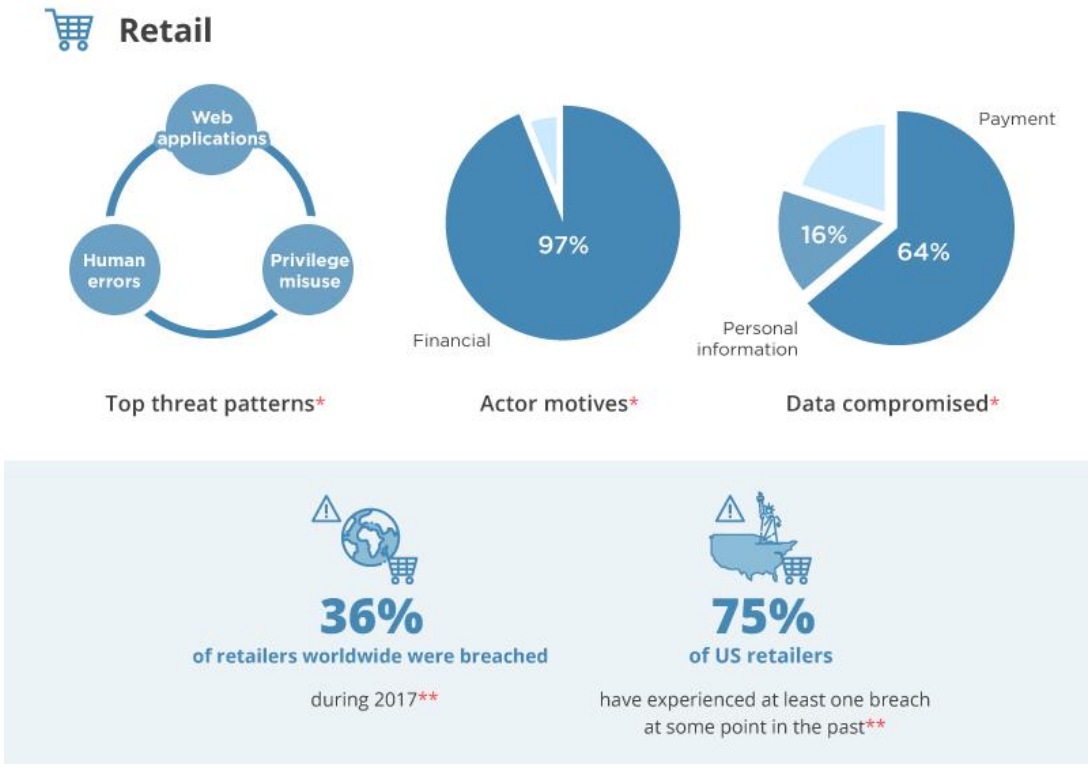

Рисунок 1.6 – Інформація стосовно витоку даних в галузі продажу [10]

Фінансові організації активно зосереджуються на впровадженні найкращих практик безпеки, які вимагаються численними галузевими стандартами. Банки постійно знаходяться під загрозою: компанії з фінансових послуг порушуються в 300 разів частіше, ніж компанії будь-якої іншої галузі. А для того, щоб проникнути в систему банківської безпеки, хакери використовують набагато складніші методи.

Більшість спроб порушення включають атаки веб-додатків. Особливо важко виявити та нейтралізувати ці атаки, оскільки мільйони клієнтів одночасно використовують ці програми.

Одне з найбільш тривожних порушень даних у фінансовому секторі сталося з Доу Джонсом. У березні 2019 року було викрито понад 2,4 мільйона записів цієї компанії. Третя сторона передала дані на загальнодоступний сервер. Ці записи містили список спостереження за ризикованими приватними особами та компаніями Dow Jones. Багато компаній використовували його для оцінки ризиків та планування роботи. . На рисунку 1.7 показано інформацію щодо сектору фінансів стосовно явища витоку даних (дані надано Verizon 2019 та Ponemon Institute 2018).

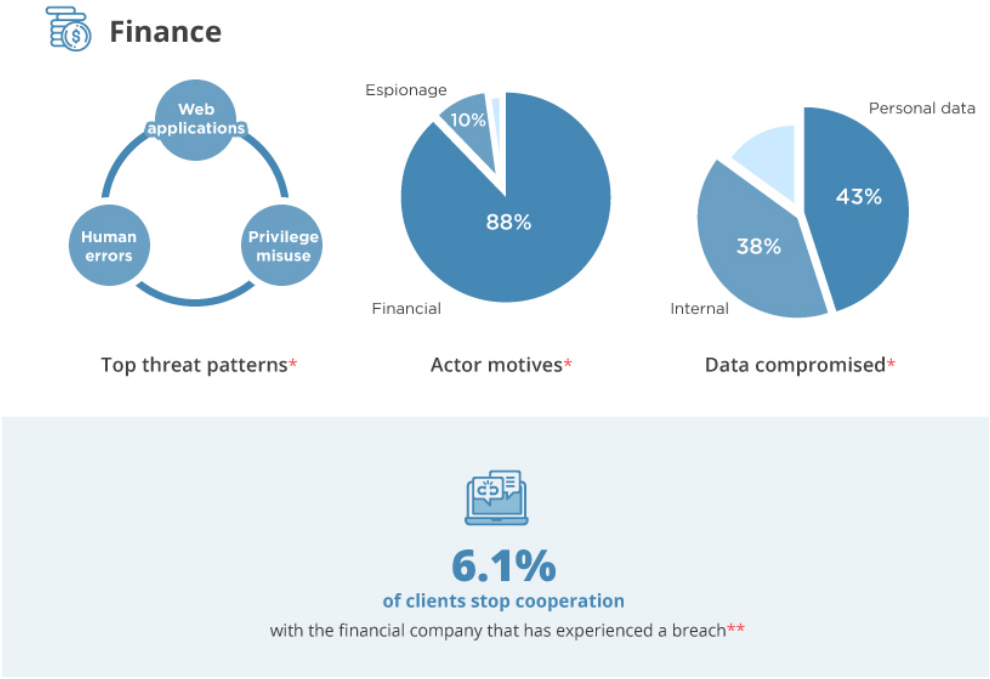

Рисунок 1.7 – Інформація стосовно витоку даних в фінансовому секторі [10]

У своєму 15-му році щорічний звіт про порушення даних, проведений Інститутом Понемона та опублікований IBM Security, продовжує надавати детальний огляд фінансових наслідків, які можуть мати інциденти безпеки для організацій, з історичними даними, що виявляють тенденції у причини та наслідки порушення даних. На рисунку 1.8 можна побачити порівняння вартості таких порушень та середній час виявлення за останні два роки.

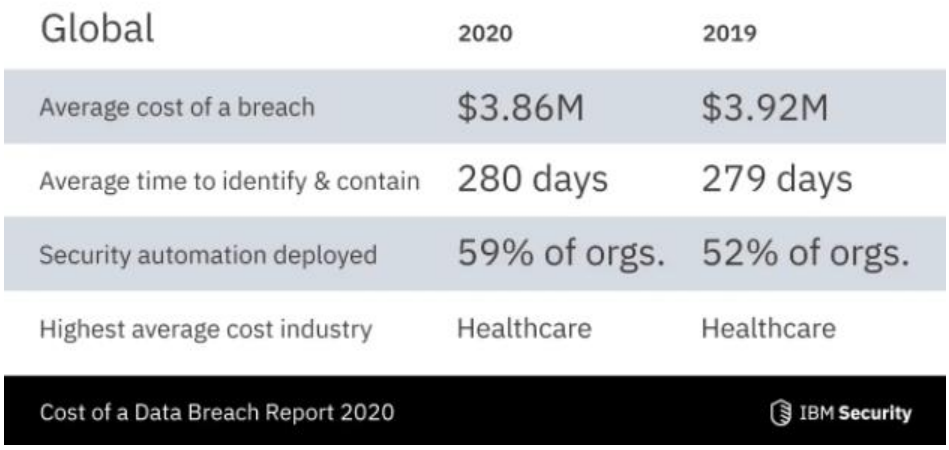

Рисунок 1.8 – Вартість витоку даних за 2019 та 2020 рік [11]

Дослідження проаналізувало 524 порушення, що відбулись у період із серпня 2019 року по квітень 2020 року в організаціях будь-якого розміру в 17 регіонах та 17 галузях. Звіт про вартість порушення даних за 2020 рік показує певну узгодженість з минулими дослідженнями, включаючи загальну вартість порушення даних, яка в дослідженні 2020 року становила в середньому 3,86 мільйона доларів США, що на 1,5% менше порівняно з дослідженням 2019 року, але відповідно до попередніх років. Середній час виявлення та утримання порушення даних становив 280 днів у дослідженні 2020 року, майже ідентичний середньому показнику 279 днів у 2019 році.

1.3. Сучасні рішення попередження витоку інформації

Для попередження витоку інформації можуть використовуватись стандартні засоби безпеки для захисту від втрати та витоку даних. Наприклад, система виявлення вторгнень (Intrusion Detection System - IDS) може попереджати про спроби зловмисників отримати доступ до конфіденційних даних. Антивірусне програмне забезпечення може перешкодити зловмисникам компрометувати важливі системи. Брандмауер може заблокувати доступ будь-якої несанкціонованої особи до систем, що зберігають конфіденційні дані.

Виявлення та запобігання вторгненню - це два широкі терміни, що описують практики захисту програм, що використовуються для пом’якшення атак та блокування нових загроз.

Перший – це пасивний захід, який ідентифікує та пом'якшує поточні атаки за допомогою системи виявлення вторгнень. Він здатний ідентифікувати відоме шкідливе програмне забезпечення (наприклад, троянські програми, бекдори, руткіти) та виявляти напади соціальної інженерії (наприклад, атака людини посередині MITM, фішинг), які маніпулюють користувачами, розкриваючи конфіденційну інформацію.

Другий – це активний захід безпеки, який використовує систему запобігання вторгненню для попереджувального блокування атак додатків. Сюди входять

віддалені включення файлів, що полегшують ін’єкції шкідливого програмного забезпечення, та ін’єкції SQL, що використовуються для доступу до баз даних підприємства.

Великі організації, можуть розглядати впровадження інструментів або повноцінних рішень DLP (Data Leakage Prevention) для захисту своїх даних. Ці інструменти можна використовувати в Центрі безпеки операцій (Security Operations Center - SOC), щоб допомогти вирішити проблему з витоком даних. Наприклад, можна використовувати систему захисту інформації та подій (SIEM) для виявлення та кореляції подій, які можуть становити витік даних.

Інструменти SIEM забезпечують:

— огляд у реальному часі в системах інформаційної безпеки організації;

— управління журналом подій, яке консолідує дані з численних джерел;

— кореляція подій, зібраних з різних журналів або джерел безпеки, використовуючи правила якщо-то (if-then), які додають логічних зав’язків до необроблених даних;

— автоматичні сповіщення про події безпеки. Більшість систем SIEM надають інформаційні панелі для питань безпеки та інших методів прямого сповіщення.

Існує багато інших дослідницьких питань та можливостей, де потрібні подальші зусилля щодо вивчення, оскільки обсяги даних на підприємстві швидко зростають. У цьому розділі виділено декілька можливих рішень.

Поглиблене навчання для виявлення внутрішньої загрози**:** у налаштуваннях big data, де генерується великий обсяг даних з різнорідних джерел, у DLPD все частіше застосовуються методи видобування даних та машинного навчання. Використовуються методи глибокого навчання, такі як Deep Neural Network для виявлення аномалій у різних додатках. Такі методи можна застосовувати як для аналізу вмісту, так і для контексту в DLPD, що зможе не тільки виявити невидимі витоки даних, але й підвищити точність і забезпечити своєчасний захист.

Машинне навчання застосовується для розв’язання задач кібербезпеки, пов’язаних з опрацюванням та аналізом великих обсягів даних: виявлення

аномальних подій, підозрілої поведінки, цільових об’єктів у великому масиві даних [12].

Поглиблене вивчення може також допомогти усунути семантичний розрив, який часто зустрічається при виявленні інсайдерських загроз. Семантичний розрив знаходиться між намірами користувача високого рівня та машинними подіями низького рівня. Наміри користувачів є найбільш доречними для виявлення інсайдерів, однак вони безпосередньо не піддаються вимірюванню. Навпаки, машинні події можна спостерігати безпосередньо, але на жаль, вони не мають сенсу і потребують відображення відповідно до намірів користувача.

DLPD як хмарний сервіс: Поява хмарних обчислень пропонує новий варіант для виявлення витоків даних. Підприємства можуть передавати свої процеси обробки даних стороннім постачальникам послуг, що викликає проблеми із конфіденційністю даних. Підхід перетину колекції заснований на подібності двох наборів з інформацією про частоту їх елементів. Отже, це може бути вразливим для частотного аналізу, якщо конфіденційні дані передаються стороннім організаціям, а третя сторона має достатню інформацію про фонову частоту n-грам. Щоб протистояти сильним атакам, необхідні алгоритми виявлення витоків даних, які зберігають конфіденційність. Важливим напрямком досліджень для постачальника хмарних послуг є досягнення масштабованості без зменшення точності виявлення та значних затримок при обробці великомасштабних наборів даних.

Моніторинг зашифрованих каналів: Більшість обговорюваних існуючих підходів DLPD вразливі до значних змін вихідних даних і, отже, не застосовуються до затуманених або зашифрованих даних. Зашифрований трафік неминуче робить виявлення на основі вмісту марним. Розгортаючи монітори зовні, зашифрований канал може частково пом'якшити проблему, майбутнє рішення DLPD потребує способу відстежувати зашифровані канали, щоб ефективно виявляти невидимі витоки даних. Можливим напрямком є використання відстеження потоків даних або диференціальний аналіз. Наприклад, нещодавно дослідники застосували техніку диференціального аналізу для досягнення стійкого виявлення витоків конфіденційності на платформах смартфонів. Збіг рядків на зашифрованих даних

був одним із найгарячіших напрямків дослідження в останнє десятиліття. Методи в цій галузі можуть також використовуватися в майбутньому DLPD для виявлення передачі конфіденційної інформації за зашифрованими каналами.

1.4. Аналіз готових рішень поставленого завдання

Перед проведенням аналізу готових рішень необхідно встановити певні вимоги до майбутньої спроектованої системи з якими в подальшому буде відбуватися порівняння.

Спроектована система повинна проводити сканування, класифікацію, аналіз та перевірку на ключові слова вхідних у неї даних. Отримувати ці дані система повинна від модулів, які дозволятимуть збір інформації з надзвичайно різних джерел інформації. Для прикладу можна навести: публікації у сервісах pastebin, публікації у чатах телеграм каналів з відповідними темами, пости з блогів ransomware груп з дарк вебу. З ціллю дотримання реальних строків створення даного проекту, буде реалізовано лише останній модуль, адже повноцінна розробка проекту подібного масштабу з великою кількістю модулів, вимагає великої кількості людино-годин.

Have I Been Pwned? - це веб-сайт, який дозволяє користувачам Інтернету перевіряти, чи не порушено їхні особисті дані. Сайт збирає та аналізує сотні дампів баз даних та публікацій, що містять інформацію про мільярди витоків облікових записів, і дозволяє користувачам шукати власну інформацію, вводячи своє ім’я користувача чи адресу електронної пошти.

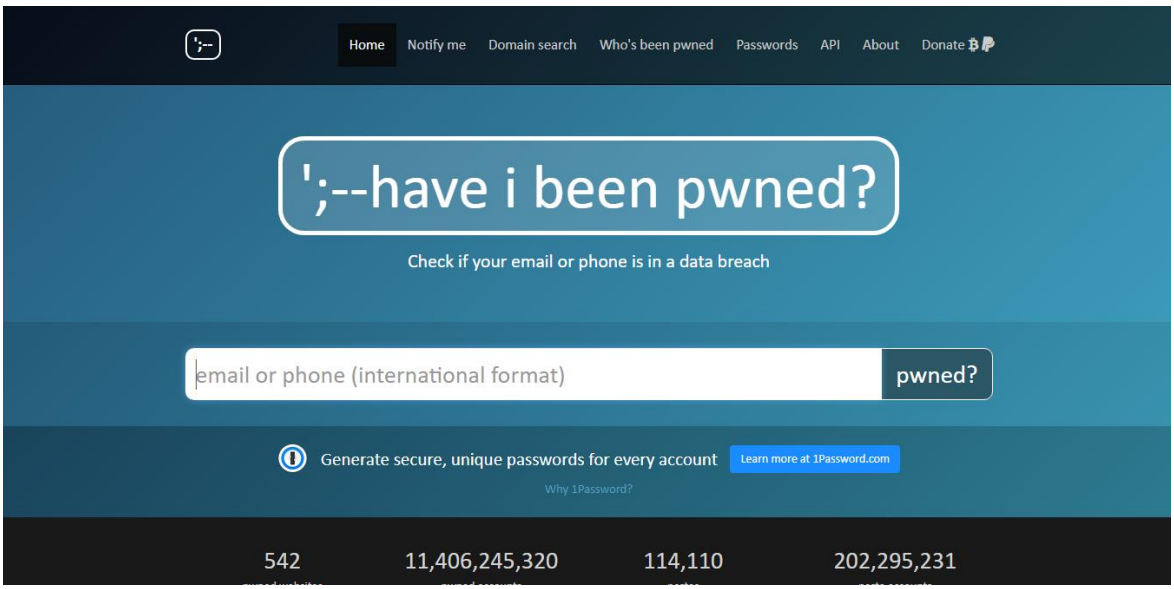

Рисунок 1.9 – Головна сторінка сервісу HIBP. [13]

Користувачі також можуть зареєструватися, щоб отримувати сповіщення, якщо їх електронна адреса з'явиться у майбутньому. Цей сайт широко рекламували як цінний ресурс для користувачів Інтернету, які бажають захистити власну безпеку та конфіденційність.

Перевагами цього сервісу є:

– Безкоштовний.

– Простий у використанні сервіс, щоб перевірити, чи не були скомпрометовані ваші дані.

– Наявність Email підписки з сповіщеннями.

До недоліків можна віднести:

– Відсутність можливості перегляду скомпрометованих паролів.

– API має не велику кількість можливостей.

– API не дає можливості безкоштовно попробувати роботу.

Intelligence X - це незалежна європейська технологічна компанія, заснована в 2018 році Пітером Кляйснером. Компанія розташована в Празі, Чеська Республіка. Його місія полягає у розробці та підтримці пошукової системи та архіву даних.

Intelligence X відрізняється від інших пошукових систем такими унікальними способами:

– Пошук працює з селекторами, тобто конкретними пошуковими термінами, такими як адреси електронної пошти, домени, URL-адреси, IP-адреси, CIDR, адреси біткойнів, хеші IPFS тощо.

– Він здійснює пошук у таких місцях, як даркнет, платформи обміну документами, Whois-дані, витоки публічних даних та інші.

– Він зберігає історичний архів даних результатів, подібно до того, як Wayback Machine з archive.org [14] зберігає історичні копії веб-сайтів.

Цільовими клієнтами даного пошукового сервісу є компанії будь-якого розміру та державні установи.

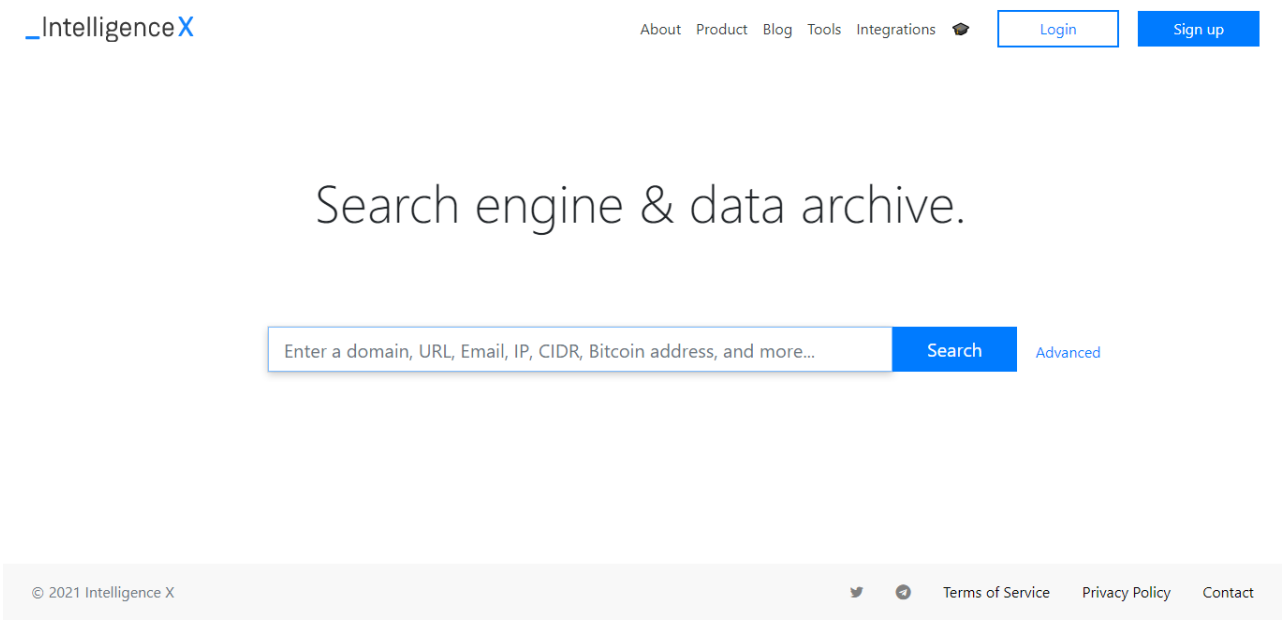

Рисунок 1.10 – Головна сторінка сервісу Intelligence X. [15]

Перевагами цього сервісу є:

– Широкий діапазон даних які можна шукати.

– Можливість розвинутої фільтрації при пошуку.

– Вивід інформації для попереднього перегляду та можливість перегляду повного джерела інформації.

– Можливість використання сервісу з студентською ліцензією безкоштовно.

До недоліків можна віднести:

- Складний інтерфейс та принцип використання.
- Мала кількість безкоштовних пошуків.
- Доступ до ексклюзивніших джерел вартує ще більше.
- Відсутність безкоштовних сповіщень про витік даних.
- Високі ціни розраховані на компанії та державні установи.

2. ОПИС ФУНКЦІОНАЛУ СИСТЕМИ ВИЯВЛЕННЯ ВИТОКІВ ДАНИХ

2.1. Характеристика системи виявлення витоку даних

Специфікацією системи є моніторинг Інтернету та збір інформації про веб-документи відповідно до уподобань користувачів. Зібрані веб-джерела даних порівнюються з конфіденційними документами користувача. Якщо в Інтернеті з'являється документ, який семантично схожий на конфіденційні документи користувачів, система вказує на можливий витік даних. На рисунку 2.1 показана абстрактна модель системи.

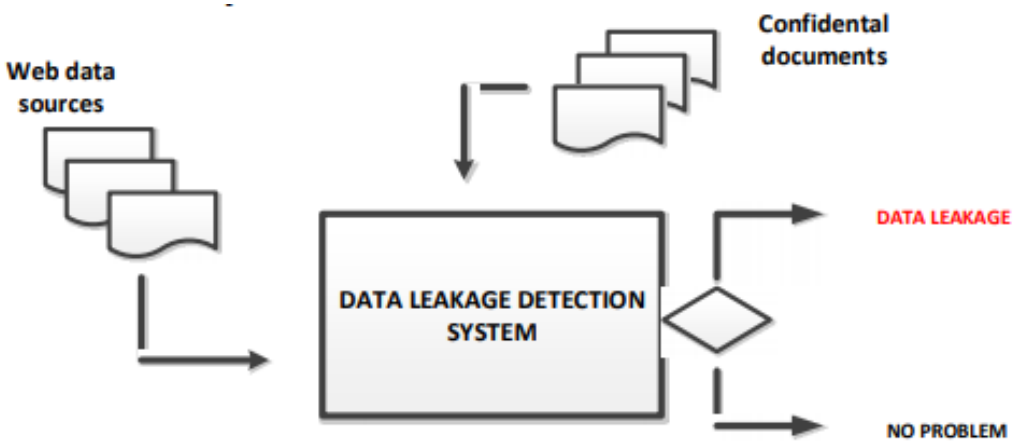

Рисунок 2.1 – Система виявлення витоку даних

Зазвичай подібність документів визначається за допомогою метрики твердої подібності на основі повторення. Цей підхід ігнорує всі потенційні семантичні кореляції

між різними словами. У системі порівнюється не чистий вміст, а значення веб-документів та документів користувачів. Система складається з ряду модулів. [16] У цьому розділі ці модулі коротко представлені. Модулі представлені на рисунку 2.2

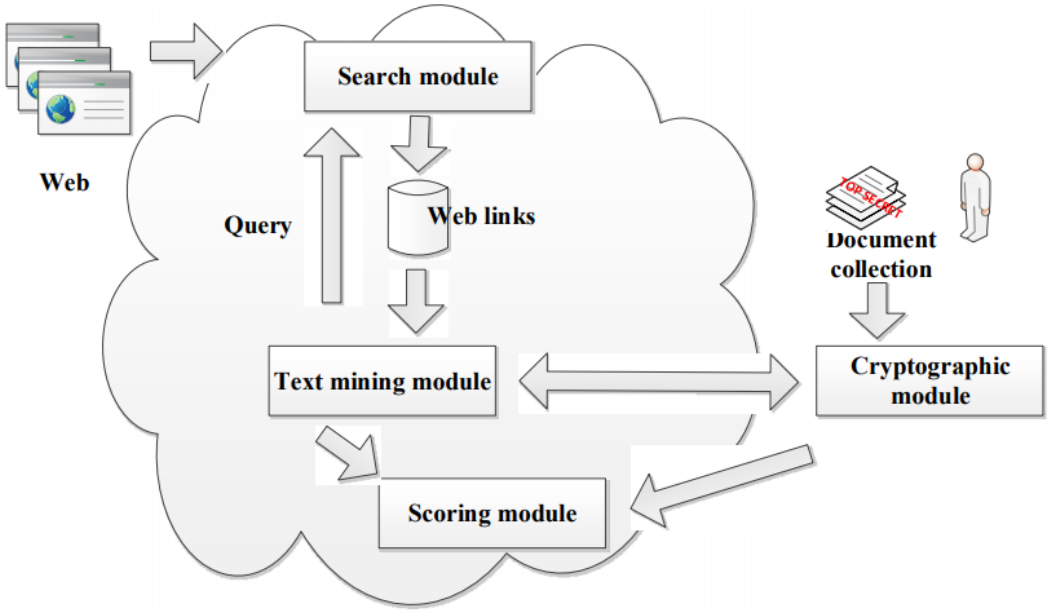

Рисунок 2.2 – Модулі системи

Колекція документів містить захищену або конфіденційну інформацію. Для захисту цих документів потрібно шифрування. Криптографічний модуль відповідає за підготовку зашифрованої версії документів. Щодо конфіденційного вмісту документів, криптографічний модуль знаходиться на сервері клієнтів. Усі інші служби розміщені в хмарі.

Модуль пошуку відповідає за виявлення веб-сторінок, які можуть свідчити про витік даних. Модуль пошуку включає модуль сканера, який досліджує структуру веб-сайтів, визначає ті сторінки веб-сайтів, які містять відповідні дані, та індексує ці сторінки за допомогою ключових слів. Модуль видобутку тексту перетворює веб-документи у відповідне математичне зображення. Модуль оцінки відповідає математичним поданням веб-документів та конфіденційних документів користувача.

2.2. Математичні методи

У цьому розділі представлені методи, які використовуються в модулі видобутку тексту, криптографії та оцінки. Враховуючи пошуковий запит, отримані веб-документи та конфіденційні документи користувачів, модуль підрахунку

обчислює оцінку відповідності, яка вимірює схожість цих документів. Модуль підрахунку балів використовує різні математичні подання документів.

Для задоволення різних інформаційних потреб або уподобань користувачів застосовуються різні математичні методи.

1. Векторна космічна модель

В ІП векторна космічна модель (VSM) широко використовується як математична модель систем. У VSM документи представлені вектором у *n*-мірному векторному просторі, де *n* - кількість термінів ключового слова чи індексу. [17]

У системі виявлення витоків даних VSM пропонується використовувати як основу модуля криптографічного та текстового аналізу. Як результат, модуль підрахунку балів може відповідати математичним представленням веб-документів та конфіденційних документів користувачів на основі векторного простору. Представлення документів VSM полягає в наступному.

З одного боку, набір ключових слів $C = \{c_1, \dots, c_i\}$ витягується з конфіденційних документів під час індексації. З іншого боку, інший набір ключових слів $W = \{w_1, \dots, w_j\}$ витягується з веб-документів під час індексації. Веб-документи та конфіденційні документи користувачів представлені за допомогою VSM над $C \cup W$ наступним чином: За умови скінченного набору $T = C \cup W$ індексних термінів $T = \{t_1, \dots, t_n\}$ будь-якому веб-документу D_j присвоюється вектор v_j кінцевих дійсних чисел, як показано нижче:

$$v_j = (w_{ij})_{i=1,\dots,n} = (w_{1j}, \dots, w_{ij}, \dots, w_{nj}) \tag{2.1}$$

Конфіденційні документи користувача U_k також повинні бути представлені у вигляді вектора v_k кінцевих дійсних чисел, як показано нижче:

$$v_k = (w_{ik})_{i=1,\dots,n} = (w_{1k}, \dots, w_{ik}, \dots, w_{nk}) \tag{2.2}$$

Вага w_{ij} інтерпретується як міра, до якої індексний термін t_i характеризує документ. Вектори веб-документів та векторів користувацьких документів порівнюються за деякою мірою подібності. Недоліком цього подання є те, що завдяки потенційній різноманітності веб-документів розмірність векторного простору може бути досить високою, що спричиняє дорогоцінні обчислення міри подібності. Веб-документ D_j представляється користувачеві, що має конфіденційний документ U_k, якщо вони досить подібні, тобто міра схожості S_{jk} між вектором веб-документа v_j та вектором конфіденційного користувача v_k перевищує деякий поріг K, тобто

$$S_{jk} = s(v_j, v_k) > K \tag{2.3}$$

У класичному VSM можуть використовуватися різні схеми зважування та міри подібності.

Індекси ваги термінів виражають, наскільки важливим є термін або ключове слово для опису змісту документа. Найчастіше використовуються такі схеми зважування. Найпростішою схемою зважування є двійкова. Це вказує на відсутність ключового слова в документі. Отже, ці ваги, як правило, недостатньо хороші, оскільки вони не розмежовують між частими та рідкісними ключовими словами. Частота зустрічальності ключових слів у документі – це кількість разів, коли це ключове слово з'являється в тексті документа. Це називається термін-частотна вага. Схема зважування на основі логарифму використовується для регулювання частоти всередині документа. Це робиться тому, що терміни, що часто з'являються в документі, не обов'язково важливіші, оскільки спеціальні терміни, які з'являються в документі лише один раз.

Розміри документів можуть сильно відрізнятися. Для компенсації цього небажаного ефекту використовуються нормалізація довжини, тобто ранг документа ділиться на його довжину. Враховуючи зважений векторний простір подання документів, їх можна порівняти, обчисливши відстань між точками, що

представляють документи. Іншими словами, використовується показник подібності, щоб документи, що мають найвищі бали, були найбільш подібними один до одного. Три типові нормовані показники подібності визначаються наступним чином [18]:

1. Подібність косинусів

$$S_{jk} = \frac{(v_j \cdot v_k)}{(\|v_j\| \cdot \| v_k\|)} \tag{2.4}$$

2. Жакардова подібність

$$S_{jk} = \frac{(v_j \cdot v_k)}{\frac{\sum_{i=1}^{n} w_{ij} + w_{ik}}{2^{w_{ij} w_{ik}}}} \tag{2.5}$$

3. Подібність кісток

$$S_{jk} = \frac{(v_j \cdot v_k)}{\frac{\sum_{i=1}^{n} w_{ij} + w_{ik}}{2^{w_{ij} w_{ik}}}} \tag{2.6}$$

Зазвичай категоричність системи може змінюватися як за рахунок зміни схеми зважування, так і міри подібності.

2. Взаємодія з інформацією

Окрім VSM, інші методи також вважаються такими, що використовують для визначення схожості документів. Для реалізації також обрано модель взаємодії інформації та пошуку (I2R). У системі виявлення витоків даних модель I2R може бути реалізована наступним чином.

Будь-який веб-документ представлений об’єктом. Кожен об'єкт o_i, $i = 1, 2, \dots, M$ асоціюється з вектором заздалегідь визначених індексних термінів $t_i = (t_{ik}), k = 1, 2, \dots, n_i$. Між будь-якою парою (o_i, o_j) $i \neq j$ встановлюються зв'язки. Посилання є зваженими та спрямованими. Дві пари посилань можна

ідентифікувати в кожному напрямку. Одне спрямоване посилання показує відносну частоту w_{ijp} індексного полінома. [19] [20] Відносна частота така:

$$w_{ijp} = \frac{r_{ijp}}{n_i}, p = 1, \dots, n_j \tag{2.7}$$

де r_{ijp} позначає доречність індексного терміна t_{jp} в об'єкті веб-документа o_i, і n_i - кількість термінів індексу в об'єкті веб-документа o_i.

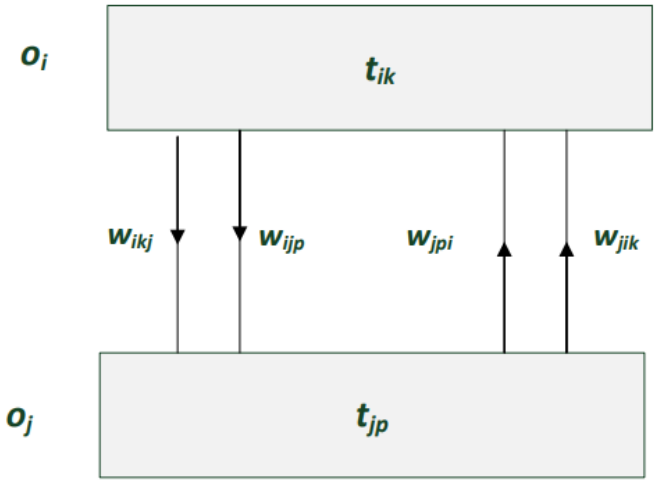

Рисунок 2.3 – Зв`язок між парами об`єктів

Іншим спрямованим посиланням є зворотна частота документа w_{ikj}. Він визначається таким чином:

$$w_{ikj} = r_{ijp} \log \frac{2M}{df_{ik}} \tag{2.8}$$

де r_{ijp} позначає відповідність індексного терміна t_{ik} в o_j, df_{ik} – кількість веб-документів, що містять ключове слово t_{ik}, а M – кількість об'єктів веб-документа. Як показано на рисунку 2.3, інший напрямок також має дві ланки з однаковим значенням. Досліджувані веб-документи представлені у вигляді повного графіку об'єктів. Конфіденційний документ користувача також представлений об’єктом. Він

пов’язаний з іншими веб-документами, які є на повному графіку. Пов’язуючи цей новий об’єкт з графіком, ваги вже існуючих зв’язків змінюватимуться. Вага пари зв'язку між будь-якою парою $(o_j, o_i), i \neq$ і, об'єктів, а отже, між конфіденційним документом та іншим об'єктом веб-документа o_i визначається як сума відповідних ваг, як показано нижче:

$$K_{ij} = \sum_{p=1}^{n_j} w_{jpi} + \sum_{k=1}^{n_j} w_{jik} \quad (2.9)$$

Цей повний графік об'єктів можна розглядати як штучну нейронну мережу. У цій мережі активація нейронів підкоряється домінанту, що приймає всі стратегії. Активація починається з конфіденційного документа користувача і поширюється на найактивніший нейрон. Після декількох кроків розповсюдження активації досягає об’єкта, на який вже відбувся вплив. Ці веб-документи схожі на конфіденційні документи користувачів, які належать до цього кола. Ці документи можуть свідчити про витік даних. Переваги моделі взаємодії полягають у наступному – з одного боку, цей метод дозволяє уникнути дорогих обчислень. Складність обчислення ваг поліноміальна. Процес також займає поліноміальний час. З іншого боку, метод пошуку взаємодії дозволяє отримати відносно високу точність в межах 50% -70%. Оцінка стандартних тестових колекцій показала, що цей метод корисний, коли надається перевага високій точності при низьких та середніх значеннях відкликання.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСОБУ АВТОМАТИЧНОГО МОНІТОРИНГУ ВИТОКУ ІНФОРМАЦІЇ

3.1. Проектування відношень компонентів автоматизованої системи

Термін автоматизованої системи являє собою організаційно-технічну систему, задача якої – обробка інформації за допомогою персоналу і комплексу засобів автоматизації. Поставлена задача автоматичного моніторингу, яка включає в себе пошук, збір, аналіз та класифікацію інформації з інтернету, являється дуже громіздкою роботою. Ідеальна система повинна мати здатність аналізувати велику кількість даних різних форматів даних які надходять з самих різних джерел мережі інтернет. Тому реалізацію даної системи було поділено на два окремих типи модулів:

– єдиний модуль аналізатора файлів – модуль який відповідає за зчитування корисних даних з широкого спектру типів файлів, а також за пошук ключових слів у цих файлах для створення передбачення про причетність файлу до витоку даних;

– модулі для пошуку та збір інформації – кількість модулів обмежується лише кількістю джерел інформації. Їх задача полягає у пошуку та збереженню файлів для подальшого аналізу.

Така модульність даного рішення дозволяє:

– виконувати їх на різних серверах;

– проводити збір та аналіз в Docker контейнерах та кластерах Kubernetes;

– можливість використання у гібридному підході розташування серверів;

– вихід одного компоненту з ладу не впливає на роботу інших компонентів.

Всі ці можливості дозволяють підтримувати максимальну стабільність роботи, дають можливість розширятись як вертикально так і горизонтально, та дозволяють економити гроші під час роботи у хмарних середовищах.

Діаграма принципу роботи наведено на рисунку 3.1.

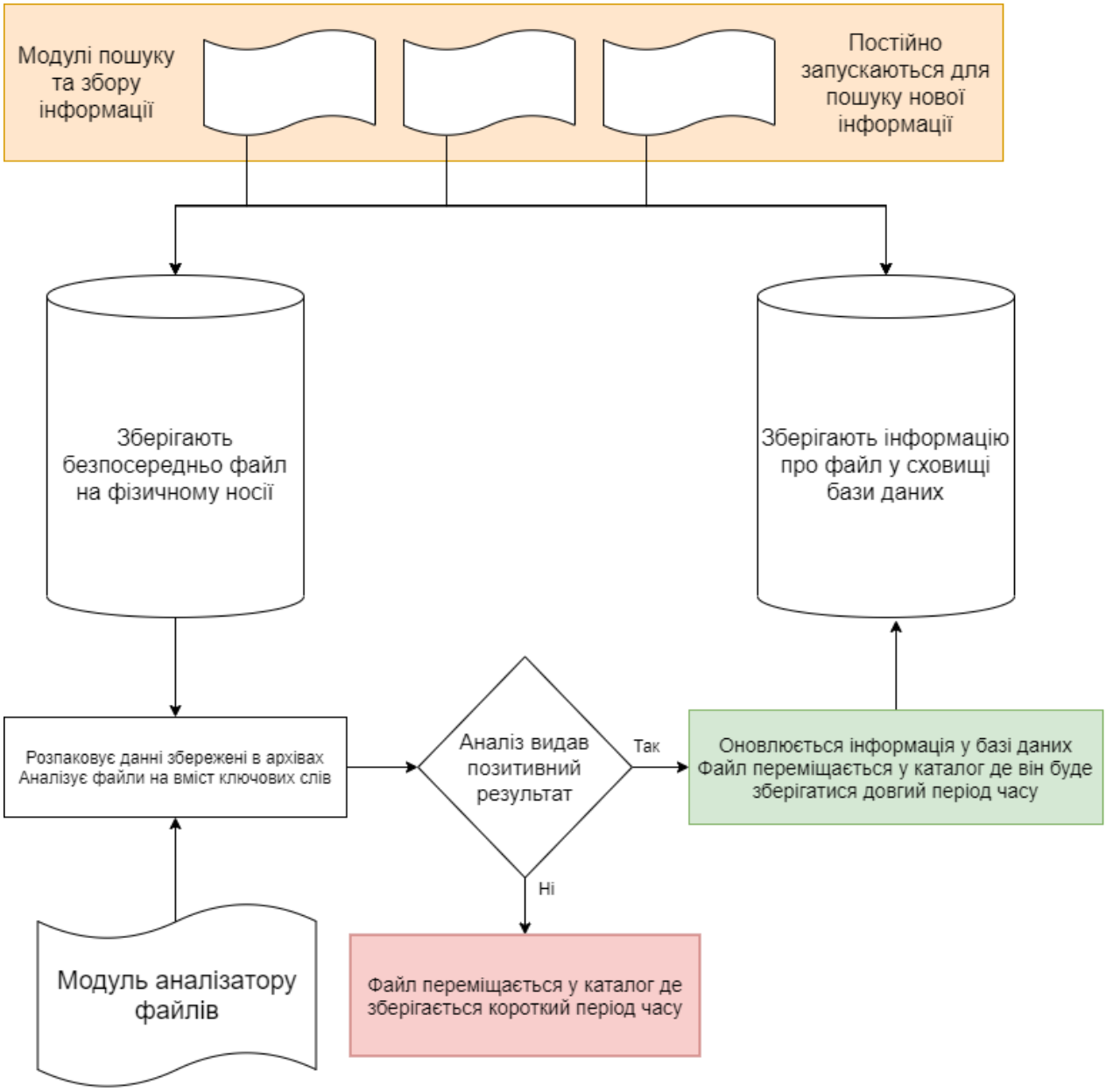

Рисунок 3.1 – Принцип роботи автоматизованої системи

Під час розробки було прийнято рішення реалізувати два мінімально необхідних компоненти, які разом дозволять побудувати систему для збору та аналізу інформації. Обов’язковим компонентом є модуль аналізатору файлів, який повинен аналізувати файли у певному каталозі. Для другого компоненту було обрано реалізувати модуль збору інформації з дар вебу.

Dark Web - це глобальна мережа, через яку користувачі можуть отримати доступ до веб ресурсів які не доступні через звичайні пошукові системи. Інформація

в Dark Web, як правило, недоступна для широких верств населення, і така інформація є навмисно прихованою від звичайного Інтернету, відомого як Clearnet.

Рішення про вибір даного джерела було обумовлене тим, що в сучасному світі все більше можна зустріти інформацію про атаки на компанії [21]. Ціллю цих атак є персональні дані компаній і у випадку якщо компанії не заплатять викуп – вся зібрана інформація стане публічно доступною.

В ході аналізу можливих джерел було обрано кілька ransomware груп, одною з яких є Conti News які налічують більше 350 компаній-жертв. Головну сторінку їх сайту можна побачити на рисунку 3.2.

Рисунок 3.2 – Головна сторінка сайту Conti News [22].

3.2. Програмна реалізація модуля пошуку та збору інформації з дарк вебу

Для доступу до інформації на сторінці в дарк вебі необхідно використовувати програмне забезпечення Tor [23]. При запуску утиліти Tor, автоматично створюється socks5h проксі через який програма буде отримати інформацію. Tor не обов'язково запускати як повноцінний браузер, для роботи достатньо запустити Tor як службу і дозволити йому працювати у фоні. Таким чином виконання даного

модуля не обмежується стаціонарними комп’ютерами, а може бути запущеним на серверах де немає графічного інтерфейсу. Для мови програмування Python існує спеціальна бібліотека Stem [24] яка дозволяє контролювати роботу сервісу Tor у фоні. Щоби не створювати проблем з роботою уже запущених служб Tor, новий запуск буде відбуватися у спеціально створеній тимчасовій директорії яка буде видалена автоматично, частину реалізації даного процесу можна побачити у лістингу 3.2, повна версія знаходиться у додатку А. Даний функціонал реалізований бібліотекою tempfile.

Лістинг 3.1 – Код запуску Tor у фоні за допомогою бібліотеки Python

```
import stem.process
from stem.control import Controller
import tempfile

class Tor_Connector():
    def __init__(self, socks_port: int, control_port: int = None, disable_init_msgs: b
ool = False):
        """
        This class is used to start Tor Instance and generate socks connection for req
uests library
        """
        self.disable_init_msgs = disable_init_msgs
        self.socks_port = socks_port
        self.control_port = control_port
        self.working_directory = tempfile.TemporaryDirectory(prefix=f'Tor_tmp_dir_port
s_{socks_port}_{control_port}_')

        if not socks_port and not control_port:
            self.launch_tor()
        else:
            config = dict()
            if self.socks_port:
                config["SocksPort"] = str(self.socks_port)
            if self.control_port:
                config["ControlPort"] = str(self.control_port)
            config["DataDirectory"] = self.working_directory.name

            self.launch_tor_with_config(config=config)

    def launch_tor_with_config(self, config: dict) -> None:
        """
        Try start Tor with config if crashes, try start Tor without one
        """
        try:
            self.tor_process = stem.process.launch_tor_with_config(config=config, init
_msg_handler=self._init_msg_handler)
        except OSError as e:
            logger.warning(f"Unable to start Tor with config, starting Tor without con
fig. Error message: {e.args[0]}")
            self.launch_tor()
```

Для роботи з інформацією на сайтах в мережі дарк веб було створено клас Deep_Web_Parser. Цей клас є батьківським і лише визначає інтерфейси реалізація яких знаходиться у дитячому класі ContiNews_Parser. Таким чином уніфіковується спосіб доступу до даних між різними парсерами сторінок, що дозволяє з легкістю добавити нове джерело не зламавши роботу програми.

Під час розробки парсеру ContiNews було знайдено незадокументований API. Після ознайомлення з принципом роботи запитів реалізація парсеру була значно спрощена, кожен з методів вимагає унікальний ідентифікатор публікації для подальшої роботи. Можна виділити два основних методи:

– get – дозволяє отримати інформацію про статтю;
– files – дозволяє отримати інформацію про файли у статі.

Після того як інформація про всі, попередньо не оброблені публікації була опрацьована та збережена у базі даних, інформація передається по сокету у модуль завантаження файлів який реалізовано у класі Watcher. Частину реалізації серверу отримання інформації можна побачити у лістингу 3.2, повну версію можна побачити у додатку Б. Цей модуль працює в багато поточному режимі, що дозволяє йому отримувати інформацію про нові публікації та завантажувати кілька файлів одночасно. Все це дуже позитивно впливає на продуктивність, так як швидкість завантаження одного файлу через мережу Tor є дуже малою, і чергу файлів на завантаження завжди можна доповнити новими файлами.

Лістинг 3.2 – Реалізація серверу отримання даних

```
def _server_connection(self):
        """
        Method is running in background infinitely and waits
        for new queue update, after receiving valid json
        data if will execute self.update_queue
        """
        # Create a TCP/IP socket
        server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        # Bind the socket to the port
        server_socket.bind((self.ip, self.port))

        # Listen for incoming connections
        server_socket.listen(1)

        while True:
            # Generate empty decoded data for every connection
```

```
        data_decoded = dict()
        # Wait for a connection
        connection, _ = server_socket.accept()
        data = ''
        try:
            # Receive the data in small chunks and append it
            while True:
                chunk = connection.recv(1024).decode(encoding="utf-8")
                if len(chunk) > 0:
                    data += chunk
                    if "END_CONVERSATION" in data:
                        data = data.replace("END_CONVERSATION", "")
                        break
                else:
                    break

            try:
                data_decoded = json.loads(data)
                resp = f"Data Received! : {data_decoded}END_CONVERSATION"
                connection.sendall(resp.encode(encoding="utf-8"))
            except Exception:
                resp = f"Could not decode data, please double check:{data}"
                logger.warning(resp)
                connection.sendall(resp.encode(encoding="utf-8"))

        finally:
            # Clean up the connection
            connection.close()

        if data_decoded:
            if data_decoded.get("action", None) == "stop_server":
                return
            if self._download_thread_started:
                self._stop_download()
            self._queue = self._generate_new_queue(data_decoded)
            self._start_download()
```

Під час завантаження для кожного файлу створюється окремий Tor проксі, це робиться для того щоби не загружати одні і ті ж ноди доступу Tor. Завдяки цьому швидкість завантаження теоретично обмежується лише можливостями обладнання на якому завантажуються файли та швидкістю яку надає інтернет провайдер. Функцію завантаження файлу наведено у лістингу 3.3, а повна реалізація модуля – у додатку Б.

Лістинг 3.3 – Функція завантаження файлу з дарк вебу

```
def _download_by_link(self, _file: dict, download_status: dict):
        # Prepearig Tor proxies
        Tor_Session = Tor_Connector(socks_port=socks_port, control_port=control_port,
disable_init_msgs=True)
        sleep(2)
        proxy = Tor_Session.get_proxy()
```

```
        # Creating requests session with params
        session = requests.Session()
        session.proxies.update(proxy)

        save_path = f"{DOWNLOAD_BASE_PATH}{source_parser}/{company_name}/{file_name}"
        while downloaded < total_length:
            resume_header = {'Range': f'bytes={downloaded}-'}
            session.headers.update(resume_header)

            res = session.get(url=url, allow_redirects=True)

            try:
                for chunk in res.iter_content(chunk_size=2048):
                    if self._download_stop_flag:
                        Tor_Session.stop_tor()

                        return 0
                    try:
                        with open(save_path, "ab+") as f:
                            f.write(chunk)
                            f.flush()
                        downloaded += len(chunk)
                        # XXX maybee add ctrl + c signal check to
                        # verify exit or to properly save data

                        with open(f"{save_path}.metadata", "w") as f_meta:
                            try:
                                f_meta.write(str(downloaded))
                                f_meta.flush()

                            except Exception as e:
                                logger.error("Cannot write len of downloaded"
                                             "data to metadata file"
                                             f"Error: {e}")
                    except Exception as e:
                        logger.error(f"Cannot write data to file Error: {e}")
                        return 1

            except Exception as e:
                logger.warning(f"Error occurred while downloading file {file_name}, pr
obably connection "
                               f"crashed, trying to continue in a moment Error: {e}")

                Tor_Session.build_new_circuit()
                sleep(5)
                continue

        download_status.pop(file_name, None)

        os.remove(f"{save_path}.metadata")

        Tor_Session.stop_tor()

        # Deleting finished download from queue. Bye bye ;)
        q_download_links = [_q_obj.get("file", {}).get("remote_location", "") for _q_o
bj in self._queue]

        self._queue.pop(q_download_links.index(url))
```

3.3. Програмна реалізація модулю аналізатора файлів

Реалізація модуля починалась з визначенням типів даних які будуть аналізуватися. У зв’язку з тим, що задача полягає у пошуку ключових слів у корпоративних документах, обов’язковим є можливість зчитування даних з таких форматів як:

- doc, docb, docm та docx;
- xls, xlsb, xlsm та xlsx;
- ppt, pptb та pptx.

Крім того було реалізовано підтримку популярних форматів даних таких як:

- pdf;
- txt;
- png;
- jpg та jpeg;
- log;
- json.

Для стандартизації роботи було створено клас Analyzer який містить реалізовані інтерфейси доступу до даних та алгоритми пошуку ключових слів у тексті, заголовку та метаданих файлів. При реалізації видобування наповнення простих текстових форматів, а також отримання метаданих з файлів була використана бібліотека textract. Функція видобування тексту з файлів наведена у лістингу 3.4.

Лістинг 3.4 – Функція для отримання вмісту файлу

```
    def get_content(self):
        try:
            return textract.process(self.file_path).decode('utf-8').lower()
        except UnicodeDecodeError:
            try:
                with open(self.file_path, 'rb') as f:
                    return f.read().decode('utf-16').lower()
            except UnicodeDecodeError:
                try:
                    with open(self.file_path, 'rb') as f:
                        return f.read().decode('utf-32').lower()
                except UnicodeDecodeError:
                    logger.error(f'[Analyzing] {self.file_path} cannot be decode in
any format (utf-8/utf-16/utf-32)')
```

```
                        self.extracting_error = "cannot be decode in any format (utf-
8/utf-16/utf-32)"

            except textract.exceptions.ExtensionNotSupported:
                self.extracting_error = 'is not supported'
                logger.warning(f'[Analyzing] {self.file_path} is not supported')
            except Exception as e:
                self.extracting_error = f'generate the following exception {e}'
                logger.error(f'[Analyzing] {self.file_path} has generate the following
exception {e}')
```

У зв’язку з тим, що часто pdf та word файли містять зображення які не можна видобути як окремий текст, було реалізовано підхід аналізу тексту, подібних документів, як зображень. Для цього файли форматів Office конвертуються у pdf за допомогою Libreoffice [25], приклад можна побачити у лістингу 3.5. Уже pdf файли конвертуються у зображення використовуючи бібліотеку pdf2image, приклад наведено у лістингу 3.6. Для видобутку тексту з зображень використовується засіб tesseract від Google, а саме обгортка для пайтону – pytesseract [26]. Приклад видобування тексту знаходиться у лістингу 3.7. Повна версія лістингів 3.5, 3.6 та 3.7 знаходиться в додатку В.

Лістинг 3.5 – Конвертування файлів форматів Office

```
self.pdffile = self.file_path.replace(self.ext, 'pdf')
os.system('soffice --headless --convert-to pdf "{}"'.format(self.file_path))

# Check if the file exists, otherwise the previous command failed
if os.path.isfile(self.pdffile):
    new_content = PdfAnalyzer(self.pdffile, 'pdf').content
    if new_content:
        if self.content:
            self.content += new_content
        else:
            self.content = new_content
    os.remove(self.pdffile)
```

Лісинг 3.6 – Конвертування pdf файлів

```
try:
    pdf = PyPDF2.PdfFileReader(self.file_path)
    logger.info(f'[INFO] Number of pages: {pdf.numPages}')
    if pdf.numPages > 100:
        logger.warning('[Warning]      analyzing      {}      can      take      more
time'.format(self.file_path))
except Exception as e:
    logger.error('Cant read number of pages for {} due to {}'.format(self.file_path,
e))

try:
```

```
            pages      =      convert_from_path(self.file_path,      300,      thread_count=4,
output_folder='/tmp',
                                        output_file=self.basename,
                                        fmt='jpeg', paths_only=True)
        except (Image.DecompressionBombError, PDFPageCountError) as e:
            logger.error('Error while deconding the pdf {}'.format(e))
            return None
```

Лістинг 3.7 – Видобування тексту з зображень

```
        for page_path in pages:
            text = str((pytesseract.image_to_string(Image.open(page_path))))
            text = text.replace('-\n', '')
            os.remove(page_path)
            content += text
```

Далі файл аналізується на вміст ключових слів, доменів, електронних адрес та номерів банківських карток, на основі чого відбувається передбачення чи причетний документ до витоку даних. Фрагмент реалізації відображений у лістингу 3.8.

Лістинг 3.8 – Реалізація передбачення витоку даних

```
        def predict_breach(report, customer_keywords):
            prediction = 0
            customer_keywords_size = len(customer_keywords)
            content_keywords_size = len(report["event"]["content"]["keywords"])
            title_keywords_size = len(report["event"]["title"]["keywords"])
            metadata_keywords_size = len(report["event"]["metadata"]["keywords"])

            content_keywords_value = 0.9 * (content_keywords_size/customer_keywords_size)
            prediction += content_keywords_value
            title_keywords_value = 0.05 * (title_keywords_size/customer_keywords_size)
            prediction += title_keywords_value
            metadata_keywords_value = 0.05 * (metadata_keywords_size/customer_keywords_size)
            prediction += metadata_keywords_value

            if len(report["event"]["content"]["emails"]) > 5:
                prediction += len(report["event"]["content"]["emails"]) * 0.02

            if len(report["event"]["content"]["credit_card"]) > 2:
                prediction += len(report["event"]["content"]["credit_card"]) * 0.02

            prediction      =      content_keywords_value      +      title_keywords_value      +
metadata_keywords_value

            return prediction
```

Результат всього аналізу зберігається у базі даних, звідки надалі інформація може бути отримана для подальшого використання.

3.4. Тестування функціональних можливостей

Після кожної розробки та впровадження нового модуля у програмний продукт, виконується функціональне тестування.

Проводиться функціональне тестування за наступними напрямками:

– регресивне тестування. Проводиться тестування розроблюваного модуля після внесення в нього змін;

– модульне тестування. Тестування окремих модулів після завершення їх розробки;

– інтеграційне тестування. Тестування правильної взаємодії модулів на правильність обробки інформації.

В процесі тестування модуля збору та зберігання інформації, за допомогою логування, було підтверджено успішне сканування, збереження інформації у базу даних Elasticsearch, та успішну передачу інформації про публікації у модуль завантаження. Результати виконання можна побачити на рисунку 3.3.

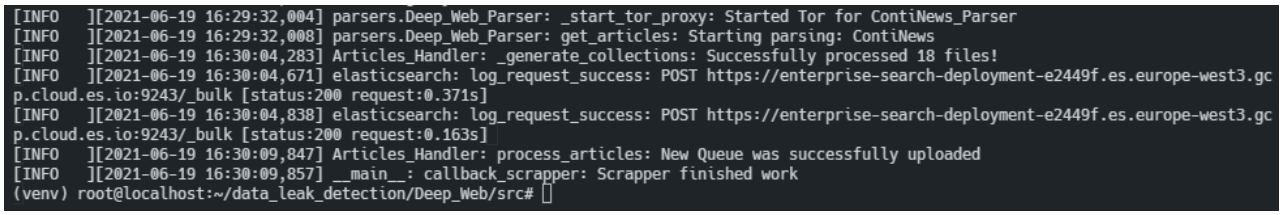

Рисунок 3.3 – Результати збору інформації про статті

Тестування модуля завантаження файлів, після отримання списку інформації про нові статті, успішно отримало додаткову інформацію про файли з бази даних, згенерувало чергу завантаження та розпочало завантаження, ця інформація відображається у логах виконання які можна побачити на рисунку 3.4.

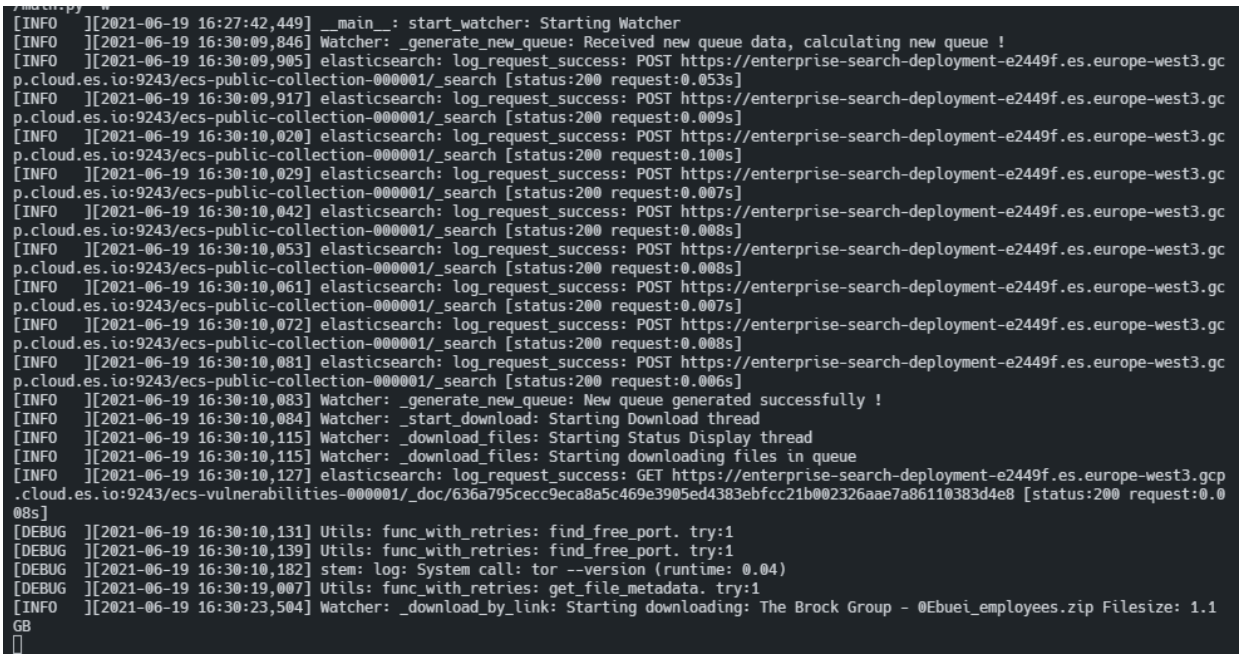

Рисунок 3.4 - Результати виконання модуля завантаження файлів

Успішність роботи модуля завантаження можна також побачити розглянувши директорію завантаження файлу. Після перевірки вмісту каталогу, через певний період, можна побачити, яку можна побачити на рисунку 3.5, що розмір файлу виріс.

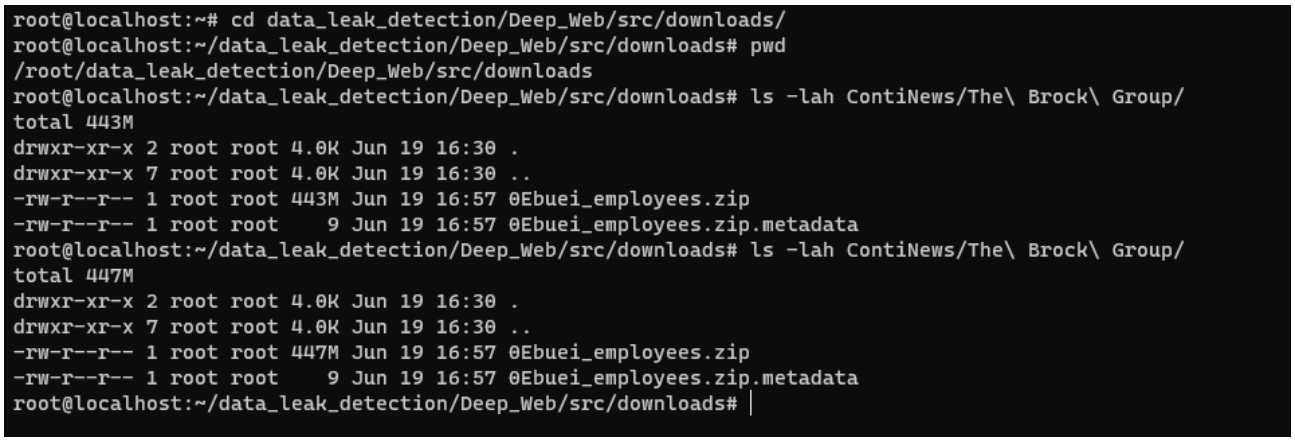

Рисунок 3.5 – Результати зміни розміру файлу під час завантаження

Після завантаження файл був переміщений у директорію для подальшого аналізу. Так як файл являється архівом, модуль аналізу розпакував файл та розпочав аналіз, що можна побачити на рисунку 3.6.

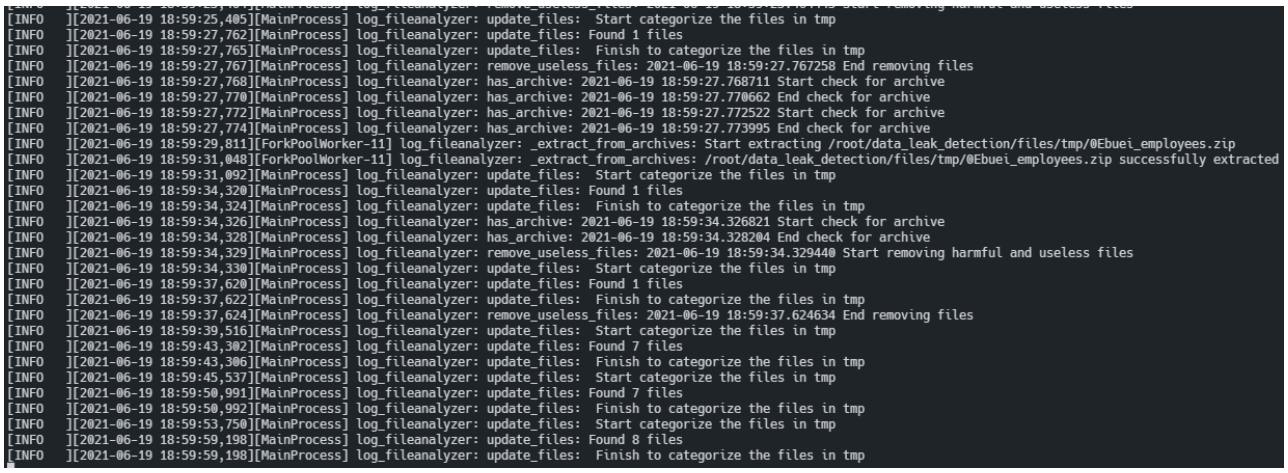

Рисунок 3.6 – Розпакування файлів та початок аналізу

Для симуляції реального прикладу витоку інформації у випадковий файл було добавлено ключові слова одного з клієнтів. Після сканування цього файлу у логах було виведено відповідне повідомлення, яке зображено на рисунку 3.7.

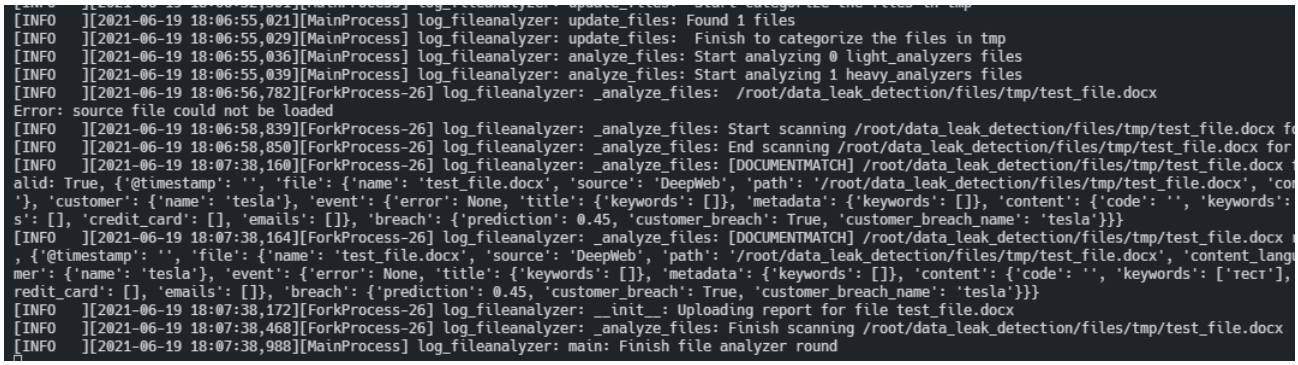

Рисунок 3.7 – Приклад сканування файлу який містить в собі ключові слова

Модульне та інтеграційне тестування компонентів системи показало відмінний результат та не виявило жодних проблем.

4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1. Долікарська допомога при ураженні електричним струмом.

При безпосередньому стиканні людини з струмоведучими частинами електроспоживачів під напругою виникає небезпека ураження її організму електрострумом, тому що тіло людини має здібність проводити електричний струм.

Важливими факторами, що визначають наслідки ураження електричним струмом, є вид струму (перемінний чи постійний), частота (при перемінному струмі), величина струму (чи напруга), тривалість дії, шлях проходження струму через тіло людини, фізичний і технічний стан людини в момент дії на його організм електричного струму (опір тіла людини). Найбільш небезпечним для людини є перемінний струм з частотою 50-500 Гц.

Величина опору різних органів тіла людини при вологій, брудній, пошкодженій шкірі різко знижається. Опір організму дії струму залежить від фізичного і психологічного стану людини і різко понижується, якщо людина голодна, нездорова, втомлена, в нетверезому стані. При цьому різко підвищюється імовірність тяжкого ураження.

Наслідок травми залежить від площі ураження і місця дотику. При одній і тій же напрузі в залежності від місця дотику в одних випадках люди зразу гинуть, в інших можуть тільки злякатися чи отримати легку травму.

З чого можна зробити висновок - необхідно постійно пам‘ятати, що електричний струм приховує в собі певну небезпеку, якщо ним невміло користуватися. Електричний струм небезпечний тим, що його дія на організм людини може викликати порушення серцевої діяльності, зупинку дихання, шоковий стан, опіки, а нерідко закінчується смертю. Внаслідок цього користування електропобутовими приладами вимагає особливої уваги та обережності від людини.

У місцях ураження електричним струмом виникають невеликі рани з твердими краями сіро-жовтого кольору, що нагадують термічний опік III ступеня. При незначній електротравмі з'являються короткочасна непритомність,

запаморочення, біль у ділянці серця, при тяжких ураженнях бувають судорожні скорочення м'язів, тривала непритомність, припиняється дихання, серцева діяльність. Шкірні покриви бліді або синюшні. У потерпілого може настати клінічна смерть. Тому врятування його життя залежить від того, наскільки швидко й ефективно буде надано першу допомогу.

Передусім слід негайно припинити дію струму, вимкнути його, якщо це можливо, або відкинути електричний провід палкою, дошкою тощо.

Після припинення дії електроструму потерпілому, навіть при легких ураженнях, потрібний спокій. При необхідності роблять штучне дихання способом «рот до рота» і непрямий масаж серця до відновлення самостійного дихання й серцевої діяльності, принаймні до прибуття на місце пригоди медичних працівників. Ділянки ураження електрострумом закривають сухою стерильною пов'язкою. Аналогічні заходи проводять при ураженні блискавкою [27].

4.2. Вимоги до виробничого освітлення та його нормування.

Серед факторів зовнішнього середовища, що впливають на організм людини в процесі праці, світло займає одне з перших місць. Адже відомо, що майже 90%. всієї інформації про довкілля людина одержує через органи зору. Під час здійснення будь-якої трудової діяльності втомлюваність очей, в основному, залежить від напруженості процесів, що супроводжують зорове сприйняття. До таких процесів відносяться адаптація, акомодація та конвергенція.

Адаптація — пристосування ока до зміни умов освітлення (рівня освітленості) [28].

Акомодація — пристосування ока до зрозумілого бачення предметів, що знаходяться від нього на неоднаковій відстані за рахунок зміни кривизни кришталика [28].

Конвергенція—здатність ока при розгляданні близьких предметів займати положення, при якому зорові осі обох очей перетинаються на предметі [28].

Яким би це не здавалося безглуздим, але і техніка безпеки на робочому місці за комп’ютером також вимагає пильної уваги до себе. Звичайно, її порушення не призведуть до трагічних наслідків, але шкоди здоров’ю неправильне обладнання робочого місця може принести.

Існує кілька факторів, випробувавши які на собі, можна зрозуміти, що пора міняти свої звички і створювати правильне робоче місце за комп’ютером. До таких факторів відносять:

- почервонілі і сльозоточиві очі в кінці робочого дня;
- печіння в очах;
- сильні головні болі;
- зниження зору;
- стомлюваність;
- дратівливість;
- безсоння;
- біль в шиї, руках і попереку.

Для створення оптимальних умов зорової роботи слід враховувати не лише кількість та якість освітлення, а й кольорове оточення. Так, при світлому пофарбуванні інтер'єру завдяки збільшенню кількості відбитого світла рівень освітленості підвищується на 20—40% (при тій же потужності джерел світла), різкість тіней зменшується, покращується рівномірність освітлення.

При надмірній яскравості джерел світла та оточуючих предметів може відбутись засліплення працівника Нерівномірність освітлення та неоднакова яскравість оточуючих предметів призводять до частої переадаптації очей під час виконання роботи і, як наслідок цього — до швидкого втомлення органів зору Тому поверхні, що добре освітлюються і знаходяться в полі зору, краще фарбувати в кольори середньої світлості, коефіцієнт відбивання яких знаходиться в межах 0,3—0,6, і, бажано, щоб вони мали матову або напівматову поверхню [28].

Для створення сприятливих умов для здорової роботи, які б запобігали швидкій втомлюваності очей, виникненню професійних захворювань, нещасних

випадків і сприяли підвищенню продуктивності праці та якості продукції, виробниче освітлення повинно відповідати наступним вимогам:

- створювати на робочій поверхні освітленість, що відповідає характеру зорової роботи і не є нижчою за встановлені норми;

- забезпечити достатню рівномірність та постійність рівня освітленості у виробничих приміщеннях, щоб уникнути частої переадаптації органів зору;

- не створювати засліплювальної дії як від самих джерел освітлення, так і від інших предметів, що знаходяться в полі зору;

- не створювати на робочій поверхні різних та глибоких тіней (особливо рухомих);

- повинен бути достатній для розрізнення деталей контраст поверхонь, що освітлюються;

- не створювати небезпечних та шкідливих виробничих чинників (шум, теплові випромінювання, небезпека уражений струмом, пожежо - та вибухонебезпека світильників):

- повинно бути надійним і простим и експлуатації, економічним та естетичним.

4.3. Вимоги пожежної безпеки при гасінні електроустановок.

Електроустановки є потенційно небезпечними місцями для виникнення пожеж, так як вони містять велику кількість горючих матеріалів і речовин (ізоляційні матеріали, масла) І потенційні джерела займання (коротке замикання, скачки напруги, перевантаження, іскри). Таке поєднання пожежонебезпечних факторів призводить до того, що саме суворе дотримання норм безпеки не може повністю усунути можливість виникнення пожежі.

Основними причинами виникнення вогнищ горіння або задимлення в електроустановках є:

– аварійні ситуації, пов'язані з перевантаженням в електромережі при відсутності захисту необхідного рівня;

– коротке замикання через пошкодження обладнання або ліній електропередач;

– несправності технологічного обладнання;

– ушкодження допоміжних електромереж;

– порушення правил експлуатації і людський фактор.

Додатковим фактором небезпеки під час пожежі в електроустановках є висока напруга - найчастіше аварійні умови не дозволяють зняти напругу на охопленому вогнем ділянці, тим більше що ситуація вимагає екстрених заходів і швидких рішень. Саме тому кожному співробітнику, задіяному в роботі на такому обладнанні, необхідно точно знати - як і чим слід гасити вогнище загоряння в електроустановках до 1000 В.

Промислові електроустановки в більшості випадків мають автоматичні засоби пожежогасіння, які починають роботу при перевищенні заданих температурних параметрів в приміщенні, аварійному відключенні електроживлення обладнання та інших факторах. При відсутності такої системи виник осередок займання або задимлення необхідно ліквідувати своїми засобами і силами до приїзду фахівців Державної служби з надзвичайних ситуацій України.

Правила пожежної безпеки України регламентують використання первинних засобів пожежогасіння на електроустановках. Згідно цих правил для гасіння електроустановок які не знаходяться під напругою можна використовувати пісок, воду і вогнегасники всіх марок. Якщо електроустановка перебуває під напругою до 1000 В - дозволено використовувати для придушення осередків займання або задимлення тільки вогнегасники порошкового, аерозольного або вуглекислотного типів з дотриманням всіх правил безпеки [29].

При виникненні вогнища загоряння в щитах управління під напругою до 400В допускається використання вуглекислотних, аерозольних або порошкових типів вогнегасників. Якщо вогнище придушити не вдається, то допускається використання розпорошених водяних потоків від протипожежного водопроводу або спеціальної техніки з обов'язковим дотриманням правил безпеки - із застосуванням

електроізолюючих рукавичок, взуття, індивідуальні засоби захисту, із заземленням пожежного ствола і насоса спецтехніки.

Під час гасіння пожежі електроустановок під напругою забороняється [30]:

– використання усіх видів піни;

– проводити будь-які відключення та інші операції з електричним обладнанням особовому складу пожежних підрозділів;

– використовувати воду зі змочувачами при подаванні компактних струменів води, як для гасіння, так і для охолодження електрообладнання та будівельних конструкцій;

– наближатися до машин і механізмів, які застосовуються для подачі води (вогнегасних речовин) на електроустановки під напругою, особам, безпосередньо не зайнятим на гасінні пожежі.

Ефективно застосовується вогнегасник, коли правильно вибрано його тип враховуючи клас пожежі, яку потрібно гасити. Вогнегасники, які містять вуглекислий газ, працюють на основі низькотемпературного струменя і відносяться до газового потоку. Після використання такого вогнегасника не залишається ніяких слідів. Однак, не слід використовувати вуглекислотні вогнегасники в замкнутому просторі, так як існує ризик пошкодження шкіри і отруєння.

Гасіння електроустановок під напругою за допомогою порошкового вогнегасника вважається ефективним методом усунення загоряння. Порошок, присутній в складі, запобігає доступ кисню до матеріалу і, отже, перешкоджає поширенню полум'я, усуваючи повторні спроби загоряння.

В інструкції до вогнегасника обов’язково міститься інформація про дату виготовлення та час проведення його останнього техобслуговування.

ВИСНОВКИ

В ході роботи було розглянуто проблему витоку інформації, її загрози, сучасні способи вирішення та проаналізовано готові рішення для вирішення цієї проблеми. Після опису функціоналу з використанням математичних моделей було спроектовано і розроблено програмне забезпечення для автоматичного моніторингу витоку інформації, яке, згодом, було протестовано та налагоджено. Функціонуюча система дає можливість:

- пошук та збір даних з джерела ContiNews розміщеного у дарк вебі;
- завантаження файлів розміщених у дарк вебі;
- розпакування файлів архівного типу;
- аналіз великої кількості форматів файлів;
- пошук ключових слів клієнтів, електронних адрес, доменів, номерів банківських карт у вмісті, заголовку чи метаданих документів;
- передбачення причетності файлу до проблеми витоку інформації.

Розробка автоматизації моніторингу витоку інформації була орієнтована на створення автономної системи по збору, аналізу та класифікації інформації. Закінчена робота являється фундаментом для потенційної системи сповіщення клієнтів та пошукового сервісу персональних даних для кінцевих користувачів. Ручний пошук витоків інформації, силами компанії або користувачів практично не можлива, так як це займало б незліченну кількість часу.

Реалізація системи проводилась з використанням мови програмування Python, а при розробці зберігалися ідеї про модульність системи, яка дозволяє окремим модулям бути запущеними на різних серверах, та навіть використовувати у гібридній архітектурі. Це дозволяє розмістити сервера для зберігання, великих у розмірі файлів, on premise, так як розміщення в хмарі було б надзвичайно дорогим через потребу в великих об’ємах носіїв інформації для зберігання даних.

Обрана мова для розробки компонентів системи Python забезпечує великий обсяг можливостей, завдяки великій кількості розширюваних пакетів. При розробці було використані бібліотеки та утиліти великих та провідних компаній таких як

Google та LibreOffice, що дає впевненість у належній та протестованій роботі ключових компонентів системи для конвертування та опрацювання файлів. Завдяки використання властивості багато поточності, дане програмне забезпечення дозволяє:

- обробляти декілька джерел одночасно;
- паралельно вести завантаження різної кількості файлів;
- розпаковувати, конвертувати та аналізувати декілька файлів одночасно.

Таким чином із збільшенням продуктивності та кількості ядер процесора зростає загальна продуктивність системи в цілому.

Для зберігання інформації про проаналізовані файли було обрано пошукову систему Elasticsearch, яка виконує роль бази даних. Рішення про вибір програмного забезпечення Elasticsearch було здійснено через потенційно велику кількість елементів пошуку, а обране ПЗ є найбільш популярним пошуковим варіантом, який може демонструвати швидкість роботи близький до реального часу навіть за умови великих об’ємів даних.

Розроблене рішення було запропоновано та впроваджено у компанії Cyberoo, де вдосконалюється, доповнюється та використовується.

В результаті розробки проекту було вдосконалено навички роботи в операційній системі Linux, розробці програм з використанням мови Python, а також досліджено та вивчено принцип роботи та пошуку Elasticsearch.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Database trends and applications. IBM's 2020 Cost of a Data Breach Study Reveals True Cost of Today's Security Glitches. Database trends and applications. 2020. URL: https://www.dbta.com/Editorial/News-Flashes/IBMs-2020-Cost-of-a-Data-Breach-Study-Reveals-True-Cost-of-Todays-Security-Glitches-142198.aspx (дата звернення: 02.04.2021).
2. Sam Smith. Business Losses to Cybercrime Data Breaches to Exceed $5 trillion by 2024. Juniper Research. 2019. URL: https://www.juniperresearch.com/press/business-losses-cybercrime-data-breaches (дата звернення: 16.04.2021).
3. BBC. Colonial Pipeline boss confirms $4.4m ransom payment. BBC. 2021. URL: https://www.bbc.com/news/business-57178503 (дата звернення: 17.04.2021).
4. F-Secure. Why do hackers want your personal information?. F-Secure. 2021. URL: https://www.f-secure.com/en/home/articles/why-do-hackers-want-your-personal-information (дата звернення: 02.05.2021).
5. Mikalauskas E. RockYou2021: largest password compilation of all time leaked online with 8.4 billion entries. Cybernews. 2021. URL: https://cybernews.com/security/rockyou2021-alltime-largest-password-compilation-leaked/ (дата звернення: 06.04.2021).
6. Drees J. Humana sued over wrongful records access data breach: 7 details. Becker's Healthcare. 2021. URL: https://www.beckershospitalreview.com/cybersecurity/humana-sued-over-wrongful-records-access-data-breach-7-details.html (дата звернення: 26.05.2021).
7. BBC News. Facebook sued over Cambridge Analytica data scandal. BBC News. 2020. URL: https://www.bbc.com/news/technology-54722362 (дата звернення: 16.05.2021).
8. Україна Сьогодні. Екс-чиновник продавав бази даних українців. Україна Сьогодні. 2017. URL: https://ukraine.segodnya.ua/ua/ukraine/eks-chinovnik-prodaval-250-baz-dannyh-ukraincev-1059322.html (дата звернення: 28.05.2021).
9. C. Long, L. Fang, Y. Danfeng. Enterprise data breach: causes, challenges,

prevention, and future directions. WIREs Data Mining and Knowledge Discovery. 2017. URL: https://wires.wiley.com/WileyCDA/WiresArticle/wisId-WIDM1211.html.

10. Ekransystem. 5 Industries Most at Risk of Data Breaches. Ekransystem. 2019. URL: https://www.ekransystem.com/en/blog/5-industries-most-risk-of-data-breaches.
11. J. Zorabedian. What's New in the 2020 Cost of a Data Breach Report. Securityintelligence. 2020. URL: https://securityintelligence.com/posts/whats-new-2020-cost-of-a-data-breach-report/ (дата звернення: 17.05.2021).
12. Застосування моделей глибокого навчання для вирішення задач кібербезпеки / О. В. Кареліна, Б. М. Липа, Р. Б. Марко, О. В. Покидко // Матеріали науково-технічної конференції „Інформаційні моделі, системи та технології“, 9-10 грудня 2020 року. — Т. : ТНТУ, 2020. — С. 37. — (Інформаційні системи та технології).
13. ';--have i been pwned?: [сайт]. [2021]. URL: https://haveibeenpwned.com/ (дата звернення: 26.05.2021).
14. Internet Archive Wayback Machine: [сайт]. [2021]. URL: https://archive.org/web/ (дата звернення: 07.05.2021).
15. Intelligence X: [сайт]. [2021]. URL: https://intelx.io/ (дата звернення: 05.06.2021).
16. A. Skrop. Data Leakage Detection Using Information Retrieval Methods. The Fourth International Conference on Advances in Information Mining and Management. 2014. С. 74-78.
17. R. Baeza-Yates, B. Ribeiro-Neto. Modern information retrieval: The Concepts and Technology behind Search (2nd Edition). ACM Press Books, 2011. С. 53-55.
18. C. T. Meadow. Text Information Retrieval Systems. Academic Press. 2000. С. 3.
19. S. Dominich. Connectionist interaction information retrieval. In: Information processing & management. 2003. С. 167-193.
20. S. Dominich. Interaction information retrieval. In: Journal of Documentation. 1994. СС. 197-212.

21. Osborne C. Colonial Pipeline attack: Everything you need to know. ZD Net. 2021. URL: https://www.zdnet.com/article/colonial-pipeline-ransomware-attack-everything-you-need-to-know/ (дата звернення: 18.05.2021).
22. Conti News: [сайт]. [2021]. URL: http:// continewsnv5otx5kaoje7krkto2qbu3gtqef22mnr7eaxw3y6ncz3ad.onion/ (дата звернення: 12.05.2021).
23. The Tor Project: [сайт]. [2021]. URL: https://www.torproject.org/ (дата звернення: 16.04.2021).
24. Welcome to Stem!: [сайт]. [2021]. URL: https://stem.torproject.org/ (дата звернення: 16.04.2021).
25. What is LibreOffice?: [сайт]. [2021]. URL: https://www.libreoffice.org/discover/ libreoffice/ (дата звернення: 01.05.2021).
26. Python-tesseract is a python wrapper for Google's Tesseract-OCR: [сайт]. [2021]. URL: https://pypi.org/project/pytesseract/ (дата звернення: 16.05.2021).
27. Скобло Ю. С, Соколовська Т. Б., Мазоренко Д. І., Тіщенко Л. М., Троянов М. М. Безпека життєдіяльності. Київ: Кондор, 2003. 352-353 pp.
28. В. Ц. Жидецький, В. С. Джигирей, О. В. Мельников. Основи охорони праці. Видання друге, стериотипне ed. Львів: Афіша, 2000. 129-146 pp.
29. Міністерство енергетики та вугільної промисловості України. Наказ, Інструкція Про затвердження Інструкції з гасіння пожеж на енергетичних об’єктах України // Відомості Верховної Ради України (ВВР). 2011. URL: https://zakon.rada.gov.ua/laws/show/z0013-12 (дата звернення: 15.05.2021).
30. Костюк В. Гасіння пожеж на електричних об’єктах під напругою // Охорона праці і пожежна безпека. 2018.
31. Alneyadi S., Sithirasenan E., та Muthukkumarasamy V., A survey on data leakage prevention systems. Journal of Network and Computer Applications, No. 62, 2016. pp. 137-152.

ДОДАТОК А

Лістинг файлу Tor_Connector.py

```
from time import sleep
import requests
import stem.process
from stem.control import Controller
from stem import Signal
import re
from json import dumps
from Utils import get_logger
import tempfile

logger = get_logger(__name__)

class Tor_Connector():
    def __init__(self, socks_port: int, control_port: int = None, disable_init_msgs: bool = False):
        """
        This class is used to start Tor Instance and generate socks connection for requests library
        """
        self.disable_init_msgs = disable_init_msgs
        self.socks_port = socks_port
        self.control_port = control_port
        self.working_directory = tempfile.TemporaryDirectory(prefix=f'Tor_tmp_dir_ports_{socks_port}_{control_port}_')
        if not socks_port and not control_port:
            self.launch_tor()
        else:
            config = dict()
            if self.socks_port:
                config["SocksPort"] = str(self.socks_port)
            if self.control_port:
                config["ControlPort"] = str(self.control_port)
            config["DataDirectory"] = self.working_directory.name

            self.launch_tor_with_config(config=config)

    def launch_tor(self) -> None:
        """
        Try start Tor without config if crashes, stop program
        """
        try:
            self.tor_process = stem.process.launch_tor(init_msg_handler=self._init_msg_handler)
        except OSError as e:
            logger.warning(f"Unable to start Tor without config, exiting. Error message: {e.args[0]}")
            raise ValueError("Unable to start Tor, please check configuration and try stoping all Tor's")

    def launch_tor_with_config(self, config: dict) -> None:
        """
        Try start Tor with config if crashes, try start Tor without one
        """
        try:
            self.tor_process = stem.process.launch_tor_with_config(config=config, init_msg_handler=self._init_msg_handler)
```

```
                except OSError as e:
                    logger.warning(f"Unable to start Tor with config, starting Tor without
config. Error message: {e.args[0]}")
                    self.launch_tor()

            def _init_msg_handler(self, line: str) -> None:
                """
                Method that catches output from Stem Tor starter parses and
                """
                if "Opened Socks listener on " in line:
                        self.proxy = self._parse_proxy(line=line)

                if not self.disable_init_msgs:
                    if "Bootstrapped " in line:
                        if re.findall(r'(Bootstrapped 0\%)', line):
                            logger.info("Starting Tor")
                        percent = re.findall(r'(?:Bootstrapped )(\d+\%)', line).pop()
                        logger.info(percent)
                    if "Bootstrapped 100% (done): Done" in line:
                        logger.info("Tor successfully started")

            def _parse_proxy(self, line: str) -> dict:
                proxy_host = re.findall(r'(?:Opened Socks listener on )([\d\.\:]+)',
line).pop()
                proxy = {
                    'http':  f'socks5h://{proxy_host}',
                    'https': f'socks5h://{proxy_host}'
                }
                if not self.disable_init_msgs:
                    logger.info("New proxy created: {}".format(dumps(proxy, sort_keys=True,
indent=4)) )
                return proxy

            def get_proxy(self) -> dict:
                return self.proxy

            def stop_tor(self) -> None:
                self.tor_process.kill()  # stops tor
                self.working_directory.cleanup()

            def check_endpoint(self) -> None:
                logger.info("Checking our endpoint:")
                # Make a request through the Tor connection
                # IP visible through Tor
                tRes = requests.get("https://api.myip.com", proxies=self.get_proxy()).json()
                logger.info("Tor IP: {}, country: {}".format(tRes.get("ip", "Error"),
tRes.get("country", "Error")))
                # Above should print an IP different than your public IP
                # Following prints your normal public IP
                logger.info("Real                                                    IP:
{}".format(requests.get("https://api.myip.com").json().get("ip", "Error")))

            def build_new_circuit(self) -> str:
                """
                Method used to build new circuit
                """
                with Controller.from_port(port = self.control_port) as controller:
                    controller.authenticate()  # provide the password here if you set one
                    controller.signal(Signal.NEWNYM)
                    sleep(controller.get_newnym_wait())
```

ДОДАТОК Б

Лістинг файлу Watcher.py

```
import json
import os
import shutil
import socket
import traceback
from functools import partial
from itertools import repeat
from multiprocessing.pool import ThreadPool
from time import sleep, time
from uuid import uuid4
import humanize as hu
import requests
from Elastic_Handler import Elastic_Handler
from Tor_Connector import Tor_Connector
from Utils import find_free_port, get_file_metadata, get_logger
logger = get_logger(__name__)
DOWNLOAD_BASE_PATH = "./downloads/"
MOVE_TO_AFTER_DL_PATH = "/root/data_leak_detection/files/tmp"
NUMBER_OF_THREADS = int(os.getenv("WATCHER_DOWNLOAD_THREADS", 1))
LOGGER_DOWNLOAD_UPDATE_TIME = 60
class Watcher():
    def __init__(self, ip: str = "0.0.0.0", port: int = 48108):
        self.ip = ip
        self.port = port
        self._queue = list()
        self._server_thread_started = False
        self._download_thread_started = False
        self._elastic_handler = Elastic_Handler()
        self._download_stop_flag = False
    def start_server(self):
        def callback(result):
            logger.info(result)
        def error_callback(exception):
            logger.error(f"Got exception in server thread. Exception: {exception}")
            raise exception
        while True:
            if not self._server_thread_started:
                self.server_thread = ThreadPool(1)
                self.server_thread.apply_async(func=self._server_connection,
                                               callback=callback,
                                               error_callback=error_callback)
                self._server_thread_started = True
                self.server_thread.close()
                self.server_thread.join()
            else:
                pass
    def _start_download(self):
        def callback(result):
            logger.info(result)
        def error_callback(exception):
            logger.error(f"Got exception in download thread. Exception: {exception}")
            logger.error(traceback.format_exception(type(exception),
                                                    exception,
                                                    exception.__traceback__))
            self._download_stop_flag = True
        if not self._download_thread_started:
```

```
            logger.info("Starting Download thread")
            self._download_thread = ThreadPool(1)
            self._download_thread.apply_async(func=self._download_files,
                                              callback=callback,
                                              error_callback=error_callback)
            self._download_thread_started = True
    def _stop_download(self):
        if self._download_thread_started:
            self._download_stop_flag = True
            self._download_thread.close()
            self._download_thread.join()
            self._download_thread_started = False
            self._download_stop_flag = False
    def _server_connection(self):
        server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        server_socket.bind((self.ip, self.port))
        server_socket.listen(1)
        while True:
            data_decoded = dict()
            connection, _ = server_socket.accept()
            data = ''
            try:
                while True:
                    chunk = connection.recv(1024).decode(encoding="utf-8")
                    if len(chunk) > 0:
                        data += chunk
                        if "END_CONVERSATION" in data:
                            data = data.replace("END_CONVERSATION", "")
                            break
                    else:
                        break
                try:
                    data_decoded = json.loads(data)
                    resp = f"Data Received! : {data_decoded}END_CONVERSATION"
                    connection.sendall(resp.encode(encoding="utf-8"))
                except Exception:
                    resp = f"Could not decode data, please double check:{data}"
                    logger.warning(resp)
                    connection.sendall(resp.encode(encoding="utf-8"))
            finally:
                connection.close()
            if data_decoded:
                if data_decoded.get("action", None) == "stop_server":
                    return
                if self._download_thread_started:
                    self._stop_download()
                self._queue = self._generate_new_queue(data_decoded)
                self._start_download()
    def _generate_new_queue(self, new_queue_data: dict):

        logger.info("Received new queue data, calculating new queue !")
        queue_backup = self._queue.copy()
        new_queue = list()
        for article_id in new_queue_data.get("default", []):
            files = self._elastic_handler.pull_files_list_by_article_id(article_id=art
icle_id)
            for _file in files:
                if _file not in queue_backup:
                    new_queue.append(_file)
        for queue_elem in queue_backup:
            if queue_elem not in new_queue:
                new_queue.append(queue_elem)
```

```
        logger.info("New queue generated successfully !")
        return new_queue
    def _download_files(self):

        with ThreadPool(processes=NUMBER_OF_THREADS + 1) as download_pool:
            def _DL_threads_CB(result, child_process: str):
                if child_process == "DOWNLOAD_THREAD":
                    pass
                elif child_process == "STATUS_THREAD":
                    pass
            def _DL_threads_ERR_CB(exception, child_process: str):
                logger.error(f"Got exception inside {child_process}. Exception: {excep
tion}")
            download_status = dict()
            iterable_args = list(zip(self._queue, repeat(download_status)))
            status_threads_CB = partial(_DL_threads_CB, child_process="STATUS_THREAD")
            status_threads_ERR_CB = partial(_DL_threads_ERR_CB, child_process="STATUS_
THREAD")
            logger.info("Starting Status Display thread")
            download_pool.apply_async(self._display_download_status,
                                      args=(download_status,),
                                      callback=status_threads_CB,
                                      error_callback=status_threads_ERR_CB)
            DL_threads_CB = partial(_DL_threads_CB, child_process="DOWNLOAD_THREAD")
            DL_threads_ERR_CB = partial(_DL_threads_ERR_CB, child_process="DOWNLOAD_TH
READ")
            logger.info("Starting downloading files in queue")

            download_pool.starmap_async(self._download_by_link,
                                        iterable=iterable_args,
                                        chunksize=1,
                                        callback=DL_threads_CB,
                                        error_callback=DL_threads_ERR_CB)
            download_pool.close()
            download_pool.join()
            return True
    def _display_download_status(self, download_status: dict) -> None:
        _logger = get_logger(__name__, output_file=False)
        while not self._download_stop_flag:
            sleep(LOGGER_DOWNLOAD_UPDATE_TIME)
            temp_dict = download_status.copy()
            _logger.info("*************DOWNLOAD STATUS*************")
            for file, status in temp_dict.items():
                _logger.info(f"File: {file} - {status}")
            _logger.info("*****************************************")
    def _download_by_link(self, _file: dict, download_status: dict):
        article = self._elastic_handler.search_article_by_file(_file)
        try:
            company_name = article.get("company_name").replace("/", "")
        except Exception as e:
            logger.error(f"Failed to extract company name from article:{article} Error
: {e}")
        try:
            source_parser = article.get("parser")
        except Exception as e:
            logger.error(f"Failed to extract parser name from article:{article} Error:
 {e}")
        try:
            url = _file.get("file", {}).get("remote_location")
        except Exception as e:
            logger.error(f"Failed to extract file url from file_dict:{_file} Error: {e
}")
```

```
        DL_directory = f'{DOWNLOAD_BASE_PATH}{source_parser}/{company_name}/'
        if not os.path.exists(DL_directory):
            try:
                os.makedirs(DL_directory)
            except Exception as e:
                logger.warning(f"Error while trying to create directory {DL_directory}
 probably it's already exist.\
                               Error: {e}")
        socks_port = find_free_port()
        control_port = find_free_port()
        Tor_Session = Tor_Connector(socks_port=socks_port, control_port=control_port,
disable_init_msgs=True)
        sleep(2)
        proxy = Tor_Session.get_proxy()
        session = requests.Session()
        session.proxies.update(proxy)
        session.stream = True
        session.verify = False
        total_length, file_name = get_file_metadata(url=url, download_session=session)
        save_path = f"{DOWNLOAD_BASE_PATH}{source_parser}/{company_name}/{file_name}"
        logger.info(f"Starting downloading: {company_name} -
 {file_name} Filesize: {hu.naturalsize(total_length)}")
        download_start_time = time()

        if os.path.exists(f"{save_path}.metadata"):
            with open(f"{save_path}.metadata", "r") as f:
                try:
                    downloaded = int(f.readline())
                except Exception:
                    open(save_path, 'w').close()
                    downloaded = 0
        else:
            downloaded = 0
        if total_length == 0:
            logger.warning(f"File `{file_name}` did not return anything")
            return 1
        while downloaded < total_length:
            resume_header = {'Range': f'bytes={downloaded}-'}
            session.headers.update(resume_header)
            res = session.get(url=url, allow_redirects=True)
            try:
                for chunk in res.iter_content(chunk_size=2048):
                    if self._download_stop_flag:
                        Tor_Session.stop_tor()
                        return 0
                    try:
                        with open(save_path, "ab+") as f:
                            f.write(chunk)
                            f.flush()
                        downloaded += len(chunk)

                        with open(f"{save_path}.metadata", "w") as f_meta:
                            try:
                                f_meta.write(str(downloaded))
                                f_meta.flush()
                            except Exception as e:
                                logger.error("Cannot write len of downloaded"
                                             "data to metadata file"
                                             f"Error: {e}")
                        DL_precent = round(downloaded / total_length * 100, 2)
                        DL_precent = f"{DL_precent}% downloaded"
                        DL_speed = downloaded // (time() - download_start_time)
```

```
                        DL_speed_str = f"Speed {hu.naturalsize(DL_speed)}/s"
                        file_size_str = f"File size {hu.naturalsize(total_length)}"
                        ES_time_int = (total_length - downloaded) // DL_speed
                        ES_time_str = hu.precisedelta(ES_time_int, minimum_unit='minut
es', format='%0.0f')
                        ES_time = f"Estimated DL time {ES_time_str}"
                        DL_status_str = f"{file_size_str} - {DL_precent} -
 {DL_speed_str} - {ES_time}"
                        download_status.update(
                            {file_name: DL_status_str})
                    except Exception as e:
                        logger.error(f"Cannot write data to file Error: {e}")
                        return 1
            except Exception as e:
                logger.warning(f"Error occurred while downloading file {file_name}, pr
obably connection "
                               f"crashed, trying to continue in a moment Error: {e}")
                Tor_Session.build_new_circuit()
                sleep(5)
                continue
        download_status.pop(file_name, None)
        os.remove(f"{save_path}.metadata")
        Tor_Session.stop_tor()

        q_download_links = [_q_obj.get("file", {}).get("remote_location", "") for _q_o
bj in self._queue]
        self._queue.pop(q_download_links.index(url))

        try:

            file_export_path = f'{MOVE_TO_AFTER_DL_PATH}DeepWeb_{uuid4()}_{file_name}'
            shutil.move(save_path, file_export_path)

        except Exception as e:
            logger.error(f"Error while trying to move the file: {e}")
        file_DL_time_used = hu.precisedelta(time() -
 download_start_time, minimum_unit='minutes', format='%0.0f')
        logger.info(f"File `{file_name}` downloaded to {file_export_path} in {file_DL_
time_used}")

        self._elastic_handler.update_downloaded_file(file_id=_file.get("id"), local_lo
cation=file_export_path)
        article_ids = list(set([_q_obj["event"]["article_id"] for _q_obj in self._queu
e]))

        if article["article_id"] not in article_ids:
            try:
                self._elastic_handler.update_downloaded_article(article_id=article["ar
ticle_id"])
            except Exception as e:
                logger.error(f"Error while trying to update downloaded article. Error:
 {e}")
            logger.info(f"All files for '{company_name}' downloaded!")
        return file_name, company_name, url
```

ДОДАТОК В

Лістинг файлу SubAnalyzer.py

```
import random
import string
from Analyzer import Analyzer
from Utils import logger
import olefile
import os
import PyPDF2
import pyexifinfo
from PIL import Image
from pdf2image import convert_from_path
from pdf2image.exceptions import PDFPageCountError
from pytesseract import pytesseract
from lxml import etree
import zipfile
from multiprocessing import Process
from timeit import default_timer as timer

# Generic file, that has not have any metadata
# It will use textextract for the content file extraction
class GenericAnalyzer(Analyzer):
    def get_metadata(self):
        return None

class OfficeAnalyzer(Analyzer):
    def __init__(self, file, filetype):
        super().__init__(file, filetype)

        # If it is a word document, convert it to pdf and then delete it
        # In this way, it provides also an OCR text extraction that could be more accu
rate, for example for images
        # presents in a word document
        if self.ext in ['doc', 'docx']:
            self.pdffile = self.file_path.replace(self.ext, 'pdf')
            os.system('soffice --headless --convert-
to pdf "{}"'.format(self.file_path))

            # Check if the file exists, otherwise the previous command failed
            if os.path.isfile(self.pdffile):
                new_content = PdfAnalyzer(self.pdffile, 'pdf').content
                if new_content:
                    if self.content:
                        self.content += new_content
                    else:
                        self.content = new_content
                os.remove(self.pdffile)

    # Get the creator and the lastModifiedBy authors
    def get_metadata(self):
        authors = ''
        if self.type[-1] == '+':
            # XML document
            try:
                zf = zipfile.ZipFile(self.file_path)
                doc = etree.fromstring(zf.read('docProps/core.xml'))
            except (zipfile.BadZipFile, KeyError) as e:
                logger.error('Unable to open the zip file due to {}'.format(e))
```

```
                return None

            ns = {'dc': 'http://purl.org/dc/elements/1.1/'}
            creator_path = doc.xpath('//dc:creator', namespaces=ns)
            if len(creator_path) > 0 and creator_path[0].text:
                authors = creator_path[0].text.lower()

            ns = {'cp': 'http://schemas.openxmlformats.org/package/2006/metadata/core-
properties'}
            creator_path = doc.xpath('//cp:lastModifiedBy', namespaces=ns)
            if len(creator_path) > 0 and creator_path[0].text:
                authors = '{} {}'.format(authors, creator_path[0].text.lower())

        elif olefile.isOleFile(self.file_path):
            # Windows 1997-2003 document
            try:
                ole = olefile.OleFileIO(self.file_path)
                meta = ole.get_metadata()
                if meta.author is not None:
                    if isinstance(meta.author, str):
                        authors = meta.author.lower()
                    else:
                        authors = meta.author.decode('utf-8').lower()

                if meta.last_saved_by is not None:
                    if isinstance(meta.last_saved_by, str):
                        authors = '{} {}'.format(authors, meta.last_saved_by.lower())
                    else:
                        authors = '{} {}'.format(authors, meta.last_saved_by.decode('u
tf-8').lower())

            except (OSError, UnicodeDecodeError):
                # print('Unable to open the document with OLE')
                return authors if authors else None

        if authors:
            logger.debug(f'{self.file_path} {authors}')

        return authors if authors else None

# Target function for MultiProcessing
# It Parses array of paths to pictures and stores result in text document
def analyze_pages(pages, thread_number, thread_secret):
    content = ''
    try:
        for page_path in pages:
            text = str((pytesseract.image_to_string(Image.open(page_path))))
            text = text.replace('-\n', '')
            os.remove(page_path)
            content += text
    except Exception as e:
        logger.error('Unable to read strings from images {} due to {}'.format(pages, e
))
        return None
    try:
        with open(f"Process{thread_number}{thread_secret}.txt", "a+") as f:
            f.write(content)
    except Exception as e:
        logger.error('Unable to store content from images {} due to {}'.format(pages,
e))
    logger.debug(f"Thread {thread_number} finished")
    return
```

```
class PdfAnalyzer(Analyzer):
    # It converts each page to an image
    # It performs the OCR text extraction on those images, that is more accurate than
reading the PDF
    # In order to save memory, it stored the .ppm file in /tmp directory
    # Before exit, delete all the images created
    # Before executing it is required to set env var OMP_THREAD_LIMIT to 2
    # You can do this with command "export OMP_THREAD_LIMIT=2" in Bash
    def get_content(self):
        import warnings
        warnings.simplefilter("ignore")
        try:
            pdf = PyPDF2.PdfFileReader(self.file_path)
            logger.info(f'[INFO] Number of pages: {pdf.numPages}')
            if pdf.numPages > 100:
                logger.warning('[Warning] analyzing {} can take more time'.format(self
.file_path))
        except Exception as e:
            logger.error('Cant read number of pages for {} due to {}'.format(self.file
_path, e))

        try:
            pages = convert_from_path(self.file_path, 300, thread_count=4, output_fold
er='/tmp',
                                      output_file=self.basename,
                                      fmt='jpeg', paths_only=True)
        except (Image.DecompressionBombError, PDFPageCountError) as e:
            logger.error('Error while deconding the pdf {}'.format(e))
            return None

        content = ''

        # Defined number of threads
        threads = 16
        os.environ["OMP_THREAD_LIMIT"] = "1"
        chunk = (len(pages)//threads) + (len(pages)/threads > 0) + 1
        tic = timer()
        processes = []
        process_number = 1
        thread_secret = ''.join(random.choice(string.ascii_lowercase) for i in range(3
0))

        # Split pages between Processes on equal chunks
        for i in range(0, len(pages), chunk):
            process = Process(target=analyze_pages, args=(pages[i:i + chunk], process_
number, thread_secret))
            processes.append(process)
            process.start()
            process_number += 1

        for process in processes:
            process.join()
        for process_number in range(1, len(processes) + 1):
            try:
                with open(f"Process{process_number}{thread_secret}.txt", "r+") as f:
                    content += f.read()
            except Exception as e:
                logger.error(f'Unable to read content from file Process{process_number
}{thread_secret}.txt due to {e}')
            try:
                with open(f"Process{process_number}{thread_secret}.txt", "r+") as f:
```

```
                    os.remove(f"Process{process_number}{thread_secret}.txt")
            except Exception as e:
                logger.error(f'Unable to remove file Process{process_number}{thread_se
cret}.txt due to {e}')

        tac = timer()
        logger.info(f"Time used for processing: {tac - tic}")

        return content.lower()

    # Get the author of the PDF
    def get_metadata(self):
        try:
            pdf = PyPDF2.PdfFileReader(self.file_path)
            info = pdf.getDocumentInfo()
        except Exception:
            return None

        return info.author.lower() if info and info.author else None

class ImageAnalyzer(Analyzer):
    # Get the GPS coordinates of the image using exif info
    def get_metadata(self):
        try:
            return pyexifinfo.information(self.file_path)
        except Exception as e:
            logger.error('Unable to get Image Metadata in {} due to {}'.format(self.fi
le_path, e))
            return None
```