The Development of automated system for monitoring the information leakage into the Darknet

Oleksandr Pokydko[a], Olena Karelina[a], Liliana Dzhydzhora[a]

[a] *Ternopil Ivan Pulyuj National Technical University, Ruska, 56, Ternopil, 46001, Ukraine*

Abstract
The system for monitoring the corporate information leakage of into the darknet is developed. The web documents collected from darknet are compared with the documents of the company-customer of monitoring and conclusion about the information leakage is made on the basis of documents similarity. The system consists of the following modules: source search module in darknet, text mining module, cryptographic module, document similarity assessment module. The following mathematical methods are used to assess the degree of similarity between corporate documents and Darknet documents: vector space model and information interaction model. Software implementation of the tool for automatic monitoring of information leakage by Python is described. The system testing results are given.

Keywords 1
Dataleakage, monitoring, Python, Tor, Darknet.

1. Introduction

Digital assets play an important or even crucial role in today's business. In all medium and large companies, payment information about transactions, agreements, customers personal data, financial documentation is formed and stored in the information system. Leakage of such information carries severe reputational and monetary losses for the company. Taking into account the peculiarities of modern business, hackers illegally access the companies files and demand a huge ransom for their unlocking and non-disclosure. It is worth mentioning the recent attacks on Colonial Pipeline, Acer, Apple, when ransoms amounted to millions of dollars.

Corporate information leakage can also be caused by insiders - employees of the company. Sometimes insiders are financially interested, sometimes the leakage is due to incompetence, inattention or mistakes in the business process. The reasons are different, but the result is the same - losses for the company. Therefore, the task of developing an automated tool for network monitoring, which would detect corporate information leakage is of great importance. If the loss of information is detected as early as possible, there are more opportunities to minimize its negative consequences. As hackers often publish information in the Darknet, the monitoring tool is developed for this purpose.

2. Background

Different aspects of information leakage monitoring are studied in scientific discourse. The solutions for cloud computing are developed in papers [1, 2]. In paper [3] the functionality of the leak detection system is based on the capabilities of the virtual machines hypervisor. In papers [4, 5] the development of information leakage monitoring systems from Android devices is revealed. Monitoring information leakage in the Darknet is an important and unresolved problem.

ITTAP'2021: 1nd InternationalWorkshop on Information Technologies: Theoretical and Applied Problems, November 16–18, 2021, Ternopil, Ukraine
EMAIL: oleksandr_pokydko2507@tntu.edu.ua (A. 1); karelina@tntu.edu.ua (A. 2), lilyadzhydzhora1970@gmail.com (A 3)
ORCID: 0000-0001-7105-7942 (A. 1); 0000-0002-5628-9048 (A. 2), 0000-0002-3672-4807 (A 3)

3. Characteristics of the data leakage detection system

The specification of the system is Darknet monitoring and collection of information about web documents in accordance with customer requirements. The collected web data sources are compared with confidential user documents. If the document that is semantically similar to users' confidential documents appears on the Internet, the system indicates possible data leakage.

Typically, the similarity of documents is determined by solid similarity metric based on repetition. This approach neglects all potential semantic correlations between different words. The system does not compare pure content, but the value of web documents and user documents. The system consists of modules presented in Figure 1.

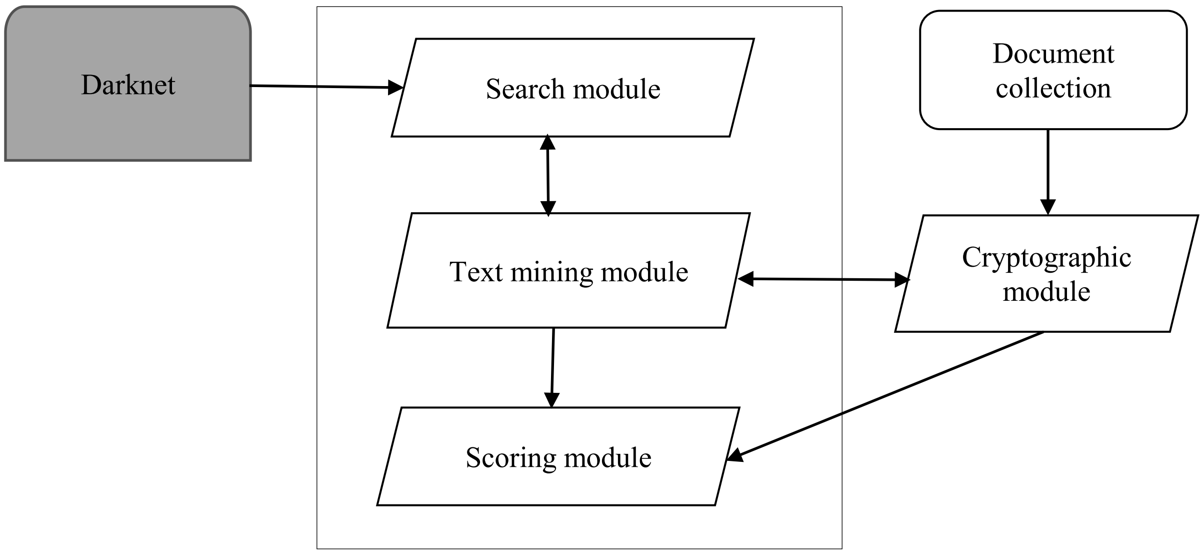


Figure 1: Modules of the system

The documents collection contains protected or confidential information. Encryption is required to protect these documents. The cryptographic module is responsible for preparing the encrypted version of the documents. To ensure the document content confidentiality, the cryptographic module is located on the client server. All other services are located in the cloud. The search module is responsible for detecting web pages that indicate data leakage. The search module includes the scanner module that examines the structure of websites, identifies those pages of websites that contain relevant data, and indexes those pages using keywords. The text extraction module converts web documents into the corresponding mathematical image. The evaluation module corresponds to the mathematical representation of web documents and confidential user documents.

4. Mathematical methods used to detect data leakage

Mathematical methods are used in the modules of text mining, cryptography and evaluation. Taking into account the search query, received web documents and confidential documents of the users, the calculation module calculates the conformity assessment, which measures the similarity of these documents. The scoring module uses various mathematical documents representations.

Depending on the type of information that prevails in the document, different mathematical methods are used.

4.1. Vector space model

Vector space model (VSM) is widely used as mathematical model of the systems. In VSM, documents are represented by the vector in n-dimensional vector space, where n is the number of terms of the keyword or index [6].

In data leakage detection system the VSM is proposed to be used as a basis for cryptographic and text analysis module. As a result, the scoring module can match the mathematical representation of web documents and user confidential documents based on vector space.

Presentation of VSM documents is as follows.

On the one hand, a set of keywords $C = \{c_1, \dots, c_i\}$ is obtained from confidential documents during indexing. On the other hand, another set of keywords $W = \{w_1, \dots, w_j\}$ is derived from web documents during indexing. Web documents and confidential user documents are represented by VSM over $C \cup W$ in the following way: under the condition of finite set $T = C \cup W$ of index terms $T = \{t_1, \dots, t_n\}$, any web document D_j is assigned vector v_j of real numbers, as shown below:

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

Confidential user documents U_k should also be presented as vector v_k of real numbers, as shown below:

$$v_k = (w_{ik})_{i=1,\dots,n} = (w_{1k}, \dots, w_{ik}, \dots, w_{nk}) \quad (2)$$

Weight w_{ij} is interpreted as the value to which the index term t_i characterizes the document. Web document vectors and customer documents vectors are compared to some degree of similarity. The disadvantage of this presentation is that due to the potential diversity of web documents, the dimension of the vector space can be quite high requiring significant computing power to determine the degree of similarity. Web document D_j is presented to the customer who has confidential document U_k, if they are quite similar, i.e. the degree of similarity S_{jk} between the web document vector v_j and the confidential user vector v_k exceeds a certain threshold, i.e.

$$S_{jk} = s(v_j, v_k) > K \quad (3)$$

Term weight indices express how important the term or keyword is to describe the content of the document. The simplest weighing scheme is binary, which indicates the presence or absence of the keyword in the document. This scheme is usually insufficient, as it does not distinguish documents with frequent and infrequent keyword inclusions. The frequency with which keywords is met in the document is called term-frequency weight. The logarithm-based weighing scheme is used to adjust the frequency within the document. This is done because the terms that are frequently met in the document are not necessarily more important than special terms that are met in the document only once.

Documents sizes can vary greatly. In order to compensate this fact, the length normalization is used, i.e. the document rank is divided by its length. Taking into account the weighted vector space of the documents submission, they can be compared by calculating the distance between the points representing the documents. In other words, the similarity indicator is used in such a way as the documents with the highest scores will be the most similar to each other.

Three typical normalized similarity indicators are defined as follows [7]:

Similarity of cosines

$$S_{jk} = \frac{(v_j \cdot v_k)}{(\|v_j\| \cdot \| v_k\|)} \quad (4)$$

Jaquard similarity

$$S_{jk} = \frac{(v_j \cdot v_k)}{\frac{\sum_{i=1}^{n} w_{ij} + w_{ik}}{2^{w_{ij} w_{ik}}}} \quad (5)$$

The system's conclusion concerning documents similarity can be changed by both changing the weighing scheme and the degree of similarity.

4.2. Information interaction model

Information interaction and search (I2R) model is also chosen to implement the information leakage monitoring system. In data leakage detection system, I2R model can be implemented in the following way. Any web document is represented by an object. Each object o_i $i = 1, 2, \dots, M$ is associated with the vector of predefined index terms $t_i = (t_{ik}), k = 1, 2, \dots, n_i$. Connections are established between any pair (o_i, o_j) $i \neq j$. Links are weighted and targeted. Two pairs of links can be identified in each direction. One directional link shows the relative frequency w_{ijp} of the index polynomial [8]. The relative frequency of the keyword in the document is defined as follows:

$$w_{ijp} = \frac{r_{ijp}}{n_i}, p = 1, \dots, n_j \tag{6}$$

where r_{ijp} indicates the relevance of the index term t_{jp} in the object of the web document o_i, n_i is the number of index terms in the web document object o_i.

Relationship between pairs of objects are schematically shown in Fig. 2.

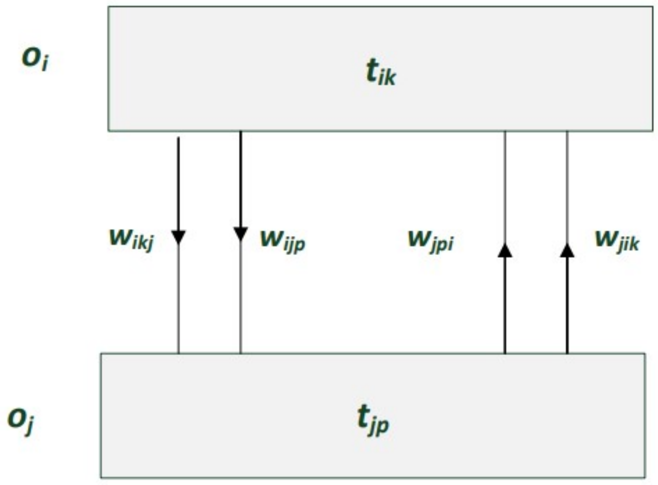


Figure 2: Relationship between pairs of objects for documents similarity determination

Another directional link is the reverse frequency of the document w_{ikj}. It is defined as follows:

$$w_{ikj} = r_{ijp} \log \frac{2M}{df_{ik}} \tag{7}$$

where r_{ijp} indicates the correspondence of the index term t_{ik} in o_j, df_{ik} is the number of web documents containing the keyword t_{ik}, and M is the number of objects of the web document. As it is shown in Figure 3, the other direction also has two links with the same value. The investigated web documents are presented in the form of complete graph of objects. The user's confidential document is also represented by the object. It is linked to other web documents that are in the full column. Linking this new object to the graph, the weights of the existing links will change. The weight of the link between any pair of objects $(o_j, o_i), i \neq \mathrm{i}$, and therefore between the confidential document and another object of the web document o_i, is defined as the sum of the corresponding weights, as shown below:

$$K_{ij} = \sum_{p=1}^{n_j} w_{jpi} + \sum_{k=1}^{n_j} w_{jik} \tag{8}$$

This complete graph of objects can be considered as an artificial neural network. In this network, the neurons activation is subjected to the dominant, which accepts all strategies. Activation begins with the user's confidential document and extends to the most active neuron. After several steps, the activation reaches the affected object. These web documents are similar to the confidential documents of users in this group. These documents can indicate the data leakage. The advantages of the interaction model are that this method avoids expensive calculations. The complexity of calculating weights is polynomial. The method of finding the interaction makes it possible to obtain relatively high accuracy within the range of 75% -85% on test collections.

5. Software implementation of the tool for automatic monitoring of information leakage

The set task of automatic monitoring of information leakage includes search, collection, analysis and classification of information from the Internet, which is very cumbersome work. The system should be able to analyze a large number of data of different formats, which come from a variety of sources on the network. Implementation of this system is presented in the form of modules:

- file analyzer module, which is responsible for reading useful data from a wide range of file types, as well as for searching for key words in these files to create a prediction about the file involvement in the source;
- modules for searching and collecting information - the number of modules is limited by the number of sources of information. Their task is to find and save files for further analysis.

Such modularity of this solution makes it possible to: perform them on different servers; conduct selection and analysis in Docker containers and Kubernetes clusters; use the hybrid approach to the location of servers; failure of one component does not affect the work of other components.

All these opportunities make it possible to maintain the maximum stability of work, give the chance to be expanded both vertically and horizontally, and enable to save money during work in cloud environments.

The scheme of operation principle of the monitoring system for information leakage is shown in Figure 3.

During the development it was decided to implement minimum two required components, which together build the system for collecting and analyzing information. The obligatory component is the file analyzer module, which should analyze the files in particular directory. For the second component the module for collecting information from Darknet is chosen for implementation.

Darknet is the global network by which users can access web resources that are not accessible through conventional search engines. Information in the Darknet is usually not available to the general public, and such information is deliberately hidden for the ordinary Internet, known as Clearnet.

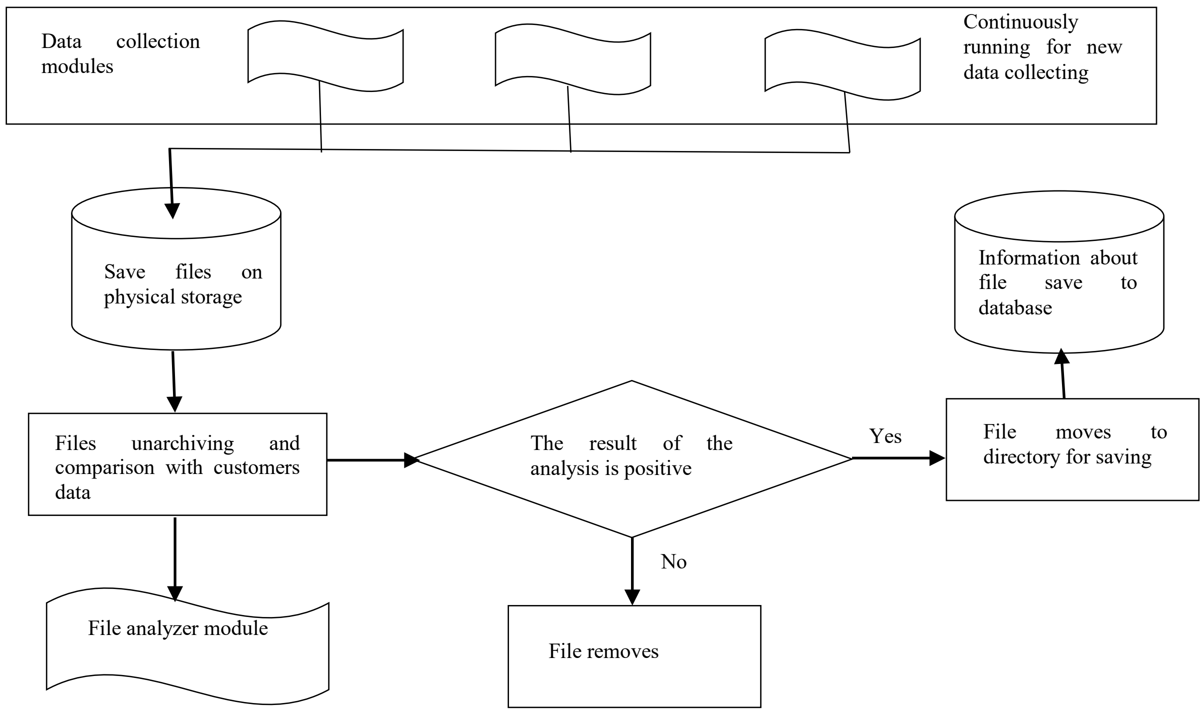

Data collection modules
Continuously running for new data collecting
Save files on physical storage
Information about file save to database
Files unarchiving and comparison with customers data
The result of the analysis is positive
Yes
File moves to directory for saving
No
File analyzer module
File removes

Figure 3. The scheme of operation principle of the information leakage monitoring system

The decision to choose this source is due to the sharp increase in the number of hacker attacks on corporate data. If companies do not pay ransom, all collected information becomes available to the public on the Darknet resource. To analyze possible sources of corporate information leakage, blogs of several ransomware-groups are selected.

To access the information on the Darknet page, you must use Tor software. When you run Tor utility, socks5h proxy is automatically created by means of which the program receive information. It is not necessary to run Tor in the browser, it is sufficient to run Tor as a service and allow it to run in the background. Thus, the implementation of this module is not limited to desktop computers, but can be run on servers where there is no graphical interface. For the Python programming language, there is a special Stem library [9], which enables you to control the operation of Tor service in the background. In order not to create problems with already running Tor services, the new launch will take place in specially created temporary directory, which will be deleted automatically at the end of the work.

The Deep_Web_Parser class is created to work with information on sites in the darknet network. This is the parent class and defines only the interfaces, the implementation of which is in the children's class ContiNews_Parser (for parsing the blog ransomware-group Conti). This unifies the method the data is accessed between different page parsers, making it possible to add easily a new source without breaking the program.

Undocumented API was found during the development of ContiNews parser. After getting acquainted with the principle of queries, the parser implementation was greatly simplified. There are two main methods of information parsing from the darknet web page:

- get – allows you to get information about the article;
- files – allows you to get information about files in the article.

After the information about all previously unprocessed publications has been processed and stored in the database, the information is passed over the socket to the file download module which is implemented in Watcher class. The part of the information retrieval server implementation can be seen in Listing 1. This module operates in multithreading mode, which makes it possible to receive information about new publications and download multiple files at once. This has a positive effect on performance, as the download speed of a single file over Tor network is very slow, and the queue for files to be downloaded can always be supplemented with new files.

Listing 1 - Data retrieval server implementation

```
    def _server_connection(self):
        """                    Methodisrunninginbackgroundinfinitelyandwaitsfornewqueueupdate,
afterreceivingvalidjson-dataifwillexecuteself.update_queue
        """
        # Create a TCP/IP socket
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        # Bindthesockettotheport
    server_socket.bind((self.ip, self.port))
        # Listenforincomingconnectionsserver_socket.listen(1)
    whileTrue:
            # Generateemptydecodeddataforeveryconnection
    data_decoded = dict()          # Waitfor a connection
    connection, _ = server_socket.accept()          data = ''          try:
              # Receivethedatainsmallchunksandappendit whileTrue:                    chunk =
connection.recv(1024).decode(encoding="utf-8")               iflen(chunk) > 0:                data
+= chunkif "END_CONVERSATION" indata:                                                      data =
data.replace("END_CONVERSATION", "")                          breakelse:
    break
    try:               data_decoded = json.loads(data)
    resp          =          f"DataReceived!          :          {data_decoded}END_CONVERSATION"
connection.sendall(resp.encode(encoding="utf-8"))               exceptException:               resp =
f"Couldnotdecodedata, pleasedoublecheck:{data}"               logger.warning(resp)
```

```
    connection.sendall(resp.encode(encoding="utf-8"))
    finally:
                # Cleanuptheconnection connection.close()
    ifdata_decoded:                          ifdata_decoded.get("action", None) == "stop_server":
returnifself._download_thread_started:               self._stop_download()
    self._queue = self._generate_new_queue(data_decoded)          self._start_download()
```

During downloading, a separate Tor-proxy is created for each file, in order not to overload the same Tor access nodes. As the result, the download speed is limited only by the capabilities of the equipment on which the files are downloaded and the speed provided by Internet-provider. The function for file downloading is shown in Listing 2.

Listing 2 - The function of file downloading from darknet

```
    def _download_by_link(self, _file: dict, download_status: dict):
        # PrepearigTorproxies
    Tor_Session    =    Tor_Connector(socks_port=socks_port,    control_port=control_port,
disable_init_msgs=True)    sleep(2)
    proxy = Tor_Session.get_proxy()         # Creatingrequestssessionwithparams session =
requests.Session()    session.proxies.update(proxy)
    save_path  =  f"{DOWNLOAD_BASE_PATH}{source_parser}/{company_name}/{file_name}"
whiledownloaded<total_length:              resume_header = {'Range': f'bytes={downloaded}-'}
session.headers.update(resume_header)

    res = session.get(url=url, allow_redirects=True)
    try:                                        forchunkinres.iter_content(chunk_size=2048):
ifself._download_stop_flag:
    Tor_Session.stop_tor()
    return 0            try:            withopen(save_path, "ab+") as f:
    f.write(chunk)
    f.flush()
    downloaded += len(chunk)
                # XXX maybeeaddctrl + c signalcheckto
                # verifyexitortoproperlysavedata
    withopen(f"{save_path}.metadata",  "w")  asf_meta:                              try:
f_meta.write(str(downloaded))              f_meta.flush()
    exceptExceptionas  e:                         logger.error("Cannotwritelenofdownloaded"
"datatometadatafile"                      f"Error:  {e}")            exceptExceptionas e:
logger.error(f"CannotwritedatatofileError: {e}")            return 1
    exceptExceptionas e:
    logger.warning(f"Erroroccurredwhiledownloadingfile {file_name}, probablyconnection "
    f"crashed, tryingtocontinuein a momentError: {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()

        # Deletingfinisheddownloadfromqueue. Byebye ;)
    q_download_links = [_q_obj.get("file", {}).get("remote_location", "") for _q_objinself._queue]
    self._queue.pop(q_download_links.index(url))
```

5.1. Software implementation of the file analyzer module

The implementation of the module starts with the definition of data types that will be analyzed. Due to the fact that the task is to find keywords in corporate documents, it is necessary to be able to read data from such formats as:

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

In addition, support for popular data formats is implemented. These formats are:

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

In order to standardize the work, Analyzer class which contains implemented data access interfaces and algorithms for searching for keywords in the text, titles and metadata files is created. Textract library is used while retrieving from plain text formats, as well as retrieving metadata from files. The function of retrieving text from files is shown in Listing 3.

Listing 3 - Function for retrieving file contents

```
defget_content(self):         try:            returntextract.process(self.file_path).decode('utf-8').lower()
exceptUnicodeDecodeError:         try:
withopen(self.file_path, 'rb') as f:
returnf.read().decode('utf-16').lower()            exceptUnicodeDecodeError:            try:
withopen(self.file_path, 'rb') as f:            returnf.read().decode('utf-32').lower()
exceptUnicodeDecodeError:
logger.error(f'[Analyzing] {self.file_path} cannotbedecodein
anyformat (utf-8/utf-16/utf-32)')
self.extracting_error = "cannotbedecodeinanyformat (utf8/utf-16/utf-32)"
excepttextract.exceptions.ExtensionNotSupported:         self.extracting_error = 'isnotsupported'
logger.warning(f'[Analyzing] {self.file_path} isnotsupported')          exceptExceptionas e:
self.extracting_error = f'generatethefollowingexception {e}'          logger.error(f'[Analyzing]
{self.file_path} hasgeneratethefollowing
exception {e}')
```

Due to the fact that pdf and word files often contain images that cannot be retrieved as separate text, the approach of text analysis from images is implemented. In order to do this, Office format files are converted to pdf using Libreoffice [25]. Then pdf files are converted into images using pdf2image library. Google's tesseract tool, i.e. pytesseract [26], is used to retrieve text from images. The example of text retrieval is given in Listing 4.

Listing 4 - Text retrieval from images

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

Further, the file is analyzed for the content of keywords, domains, email addresses and bank card numbers, on the basis of this analysis the prediction if the document refers to data leakage is made. The implementation fragment is shown in Listing 5.

Listing 5 - Implementation of data leakage prediction

```
defpredict_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
    iflen(report["event"]["content"]["emails"]) > 5:
    prediction += len(report["event"]["content"]["emails"]) * 0.02
    iflen(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
    returnprediction
```

The result of the entire analysis is stored in the database, where the information can be obtained for further application.

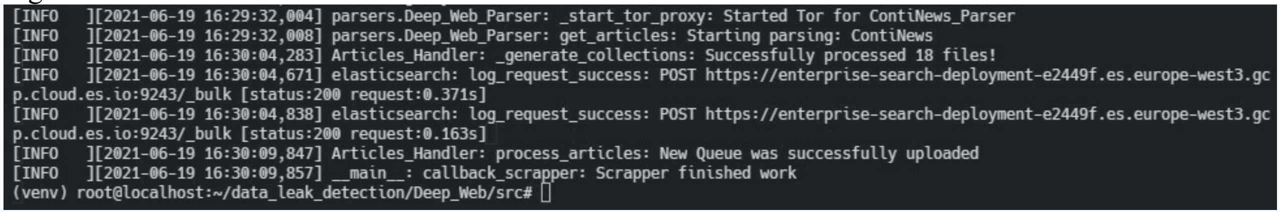
5.2. Testing the functionality of the information leakage monitoring system

After the development and implementation of the new module in software product, functional testing is carried out.

Functional testing is carried out in the following areas:

- regression testing. Testing of the developed module after changes in it is carried out;
- modular testing. Testing of separate modules after their development;
- integration testing. Testing the correct modules interaction for information processing correctness.

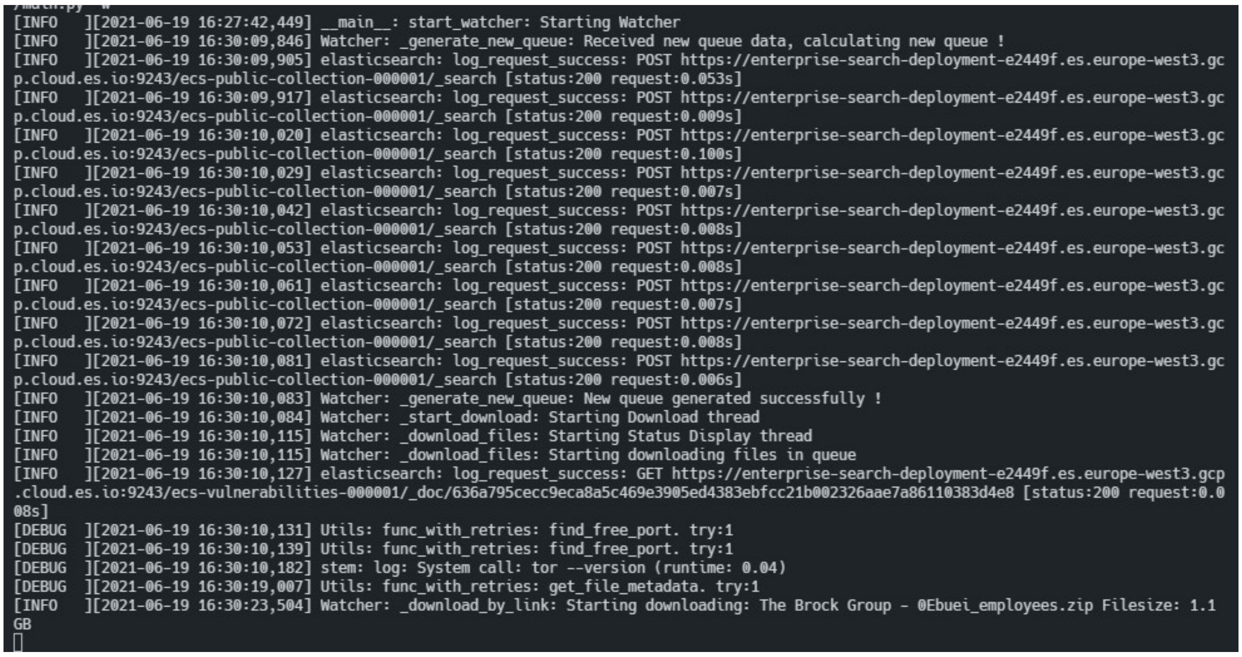
In the process of testing the module for collecting and storing information by logging the successful scanning, information storage to Elasticsearch database, and successful transfer of information about publications to the download module are confirmed. The results are shown in Figure 4.



```
[INFO   ][2021-06-19 16:29:32,004] parsers.Deep_Web_Parser: _start_tor_proxy: Started Tor for ContiNews_Parser
[INFO   ][2021-06-19 16:29:32,008] parsers.Deep_Web_Parser: get_articles: Starting parsing: ContiNews
[INFO   ][2021-06-19 16:30:04,283] Articles_Handler: _generate_collections: Successfully processed 18 files!
[INFO   ][2021-06-19 16:30:04,671] elasticsearch: log_request_success: POST https://enterprise-search-deployment-e2449f.es.europe-west3.gc
p.cloud.es.io:9243/_bulk [status:200 request:0.371s]
[INFO   ][2021-06-19 16:30:04,838] elasticsearch: log_request_success: POST https://enterprise-search-deployment-e2449f.es.europe-west3.gc
p.cloud.es.io:9243/_bulk [status:200 request:0.163s]
[INFO   ][2021-06-19 16:30:09,847] Articles_Handler: process_articles: New Queue was successfully uploaded
[INFO   ][2021-06-19 16:30:09,857] __main__: callback_scrapper: Scrapper finished work
(venv) root@localhost:~/data_leak_detection/Deep_Web/src#
```

Figure 4: The results of collecting information about the article

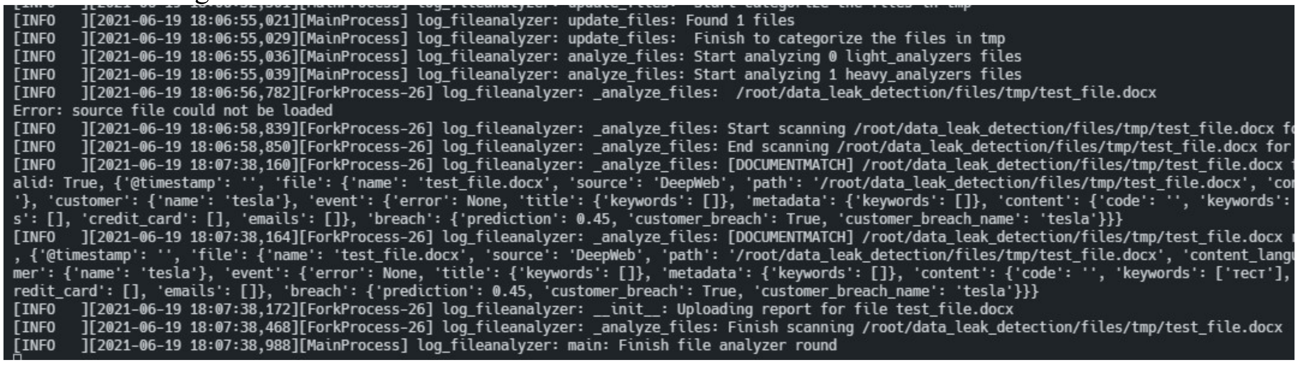
The test of file download module, after receiving the list of information about new articles, successfully obtains additional information about files from the database, generates the download queue and starts downloading, this information is displayed in the execution logs, which is presented in Figure 5.



```
[INFO   ][2021-06-19 16:27:42,449] __main__: start_watcher: Starting Watcher
[INFO   ][2021-06-19 16:30:09,846] Watcher: _generate_new_queue: Received new queue data, calculating new queue !
[INFO   ][2021-06-19 16:30:09,905] elasticsearch: log_request_success: POST https://enterprise-search-deployment-e2449f.es.europe-west3.gc
p.cloud.es.io:9243/ecs-public-collection-000001/_search [status:200 request:0.053s]
[INFO   ][2021-06-19 16:30:09,917] elasticsearch: log_request_success: POST https://enterprise-search-deployment-e2449f.es.europe-west3.gc
p.cloud.es.io:9243/ecs-public-collection-000001/_search [status:200 request:0.009s]
[INFO   ][2021-06-19 16:30:10,020] elasticsearch: log_request_success: POST https://enterprise-search-deployment-e2449f.es.europe-west3.gc
p.cloud.es.io:9243/ecs-public-collection-000001/_search [status:200 request:0.100s]
[INFO   ][2021-06-19 16:30:10,029] elasticsearch: log_request_success: POST https://enterprise-search-deployment-e2449f.es.europe-west3.gc
p.cloud.es.io:9243/ecs-public-collection-000001/_search [status:200 request:0.007s]
[INFO   ][2021-06-19 16:30:10,042] elasticsearch: log_request_success: POST https://enterprise-search-deployment-e2449f.es.europe-west3.gc
p.cloud.es.io:9243/ecs-public-collection-000001/_search [status:200 request:0.008s]
[INFO   ][2021-06-19 16:30:10,053] elasticsearch: log_request_success: POST https://enterprise-search-deployment-e2449f.es.europe-west3.gc
p.cloud.es.io:9243/ecs-public-collection-000001/_search [status:200 request:0.008s]
[INFO   ][2021-06-19 16:30:10,061] elasticsearch: log_request_success: POST https://enterprise-search-deployment-e2449f.es.europe-west3.gc
p.cloud.es.io:9243/ecs-public-collection-000001/_search [status:200 request:0.007s]
[INFO   ][2021-06-19 16:30:10,072] elasticsearch: log_request_success: POST https://enterprise-search-deployment-e2449f.es.europe-west3.gc
p.cloud.es.io:9243/ecs-public-collection-000001/_search [status:200 request:0.008s]
[INFO   ][2021-06-19 16:30:10,081] elasticsearch: log_request_success: POST https://enterprise-search-deployment-e2449f.es.europe-west3.gc
p.cloud.es.io:9243/ecs-public-collection-000001/_search [status:200 request:0.006s]
[INFO   ][2021-06-19 16:30:10,083] Watcher: _generate_new_queue: New queue generated successfully !
[INFO   ][2021-06-19 16:30:10,084] Watcher: _start_download: Starting Download thread
[INFO   ][2021-06-19 16:30:10,115] Watcher: _download_files: Starting Status Display thread
[INFO   ][2021-06-19 16:30:10,115] Watcher: _download_files: Starting downloading files in queue
[INFO   ][2021-06-19 16:30:10,127] elasticsearch: log_request_success: GET https://enterprise-search-deployment-e2449f.es.europe-west3.gcp
.cloud.es.io:9243/ecs-vulnerabilities-000001/_doc/636a795cecc9eca8a5c469e3905ed4383ebfcc21b002326aae7a86110383d4e8 [status:200 request:0.0
08s]
[DEBUG  ][2021-06-19 16:30:10,131] Utils: func_with_retries: find_free_port. try:1
[DEBUG  ][2021-06-19 16:30:10,139] Utils: func_with_retries: find_free_port. try:1
[DEBUG  ][2021-06-19 16:30:10,182] stem: log: System call: tor --version (runtime: 0.04)
[DEBUG  ][2021-06-19 16:30:19,007] Utils: func_with_retries: get_file_metadata. try:1
[INFO   ][2021-06-19 16:30:23,504] Watcher: _download_by_link: Starting downloading: The Brock Group - 0Ebuei_employees.zip Filesize: 1.1
GB
```

Figure 5: The results of file download module performance

In order to simulate the real example of information leakage, the keywords of one of the customers are added to random file. After this file scanning, the corresponding message is displayed in the logs. It is shown in Figure 6.



```
[INFO   ][2021-06-19 18:06:55,021][MainProcess] log_fileanalyzer: update_files: Found 1 files
[INFO   ][2021-06-19 18:06:55,029][MainProcess] log_fileanalyzer: update_files:  Finish to categorize the files in tmp
[INFO   ][2021-06-19 18:06:55,036][MainProcess] log_fileanalyzer: analyze_files: Start analyzing 0 light_analyzers files
[INFO   ][2021-06-19 18:06:55,039][MainProcess] log_fileanalyzer: analyze_files: Start analyzing 1 heavy_analyzers files
[INFO   ][2021-06-19 18:06:56,782][ForkProcess-26] log_fileanalyzer: _analyze_files:  /root/data_leak_detection/files/tmp/test_file.docx
Error: source file could not be loaded
[INFO   ][2021-06-19 18:06:58,839][ForkProcess-26] log_fileanalyzer: _analyze_files: Start scanning /root/data_leak_detection/files/tmp/test_file.docx fo
[INFO   ][2021-06-19 18:06:58,850][ForkProcess-26] log_fileanalyzer: _analyze_files: End scanning /root/data_leak_detection/files/tmp/test_file.docx for
[INFO   ][2021-06-19 18:07:38,160][ForkProcess-26] log_fileanalyzer: _analyze_files: [DOCUMENTMATCH] /root/data_leak_detection/files/tmp/test_file.docx
alid: True, {'@timestamp': '', 'file': {'name': 'test_file.docx', 'source': 'DeepWeb', 'path': '/root/data_leak_detection/files/tmp/test_file.docx', 'co
'}, 'customer': {'name': 'tesla'}, 'event': {'error': None, 'title': {'keywords': []}, 'metadata': {'keywords': []}, 'content': {'code': '', 'keywords':
s': [], 'credit_card': [], 'emails': []}, 'breach': {'prediction': 0.45, 'customer_breach': True, 'customer_breach_name': 'tesla'}}}
[INFO   ][2021-06-19 18:07:38,164][ForkProcess-26] log_fileanalyzer: _analyze_files: [DOCUMENTMATCH] /root/data_leak_detection/files/tmp/test_file.docx
, {'@timestamp': '', 'file': {'name': 'test_file.docx', 'source': 'DeepWeb', 'path': '/root/data_leak_detection/files/tmp/test_file.docx', 'content_langu
mer': {'name': 'tesla'}, 'event': {'error': None, 'title': {'keywords': []}, 'metadata': {'keywords': []}, 'content': {'code': '', 'keywords': ['тест'],
redit_card': [], 'emails': []}, 'breach': {'prediction': 0.45, 'customer_breach': True, 'customer_breach_name': 'tesla'}}}
[INFO   ][2021-06-19 18:07:38,172][ForkProcess-26] log_fileanalyzer: __init__: Uploading report for file test_file.docx
[INFO   ][2021-06-19 18:07:38,468][ForkProcess-26] log_fileanalyzer: _analyze_files: Finish scanning /root/data_leak_detection/files/tmp/test_file.docx
[INFO   ][2021-06-19 18:07:38,988][MainProcess] log_fileanalyzer: main: Finish file analyzer round
```

Figure 6: The example of scanning the file containing customer's keywords

Modular and integration testing of the system components showed excellent results and did not reveal any problems.

6. Conclusions and directions for further investigations

The developed system of automatic monitoring of corporate information leakage makes it possible to reveal the corporate information leakage as soon as possible and to minimize its consequences. At present the problem of hacker attacks on companies is very important all over the world. Different countries are taking measures to protect businesses from attacks by intruders in the digital space. However, the companies should take care of data security themselves.

The proposed development will be expanded in order to cover new sources of information (hacker forums, shadow Telegram channels, etc.). For some companies (design, design bureaus) graphic information is of primary importance. In further research it is necessary to investigate mathematical methods of similarity detection for graphic, audio, video files.

7. References

[1] R. Kirdat, N. Mokal, J. Mokal, A. Parkar, R. Shahabade. “Data Leakage Detection and File Monitoring in Cloud Computing” Internetional Journal of Advance Research, Ideas and Innovations in Technology. Volume 4, Issue 2, 2018. pp. 859-866.

[2] R. Naik and M. N. Gaonkar, "Data Leakage Detection in cloud using Watermarking Technique," 2019 International Conference on Computer Communication and Informatics (ICCCI), 2019, pp. 1-6, doi: 10.1109/ICCCI.2019.8821894.

[3] Chang, S.-H., Mallissery, S., Hsieh, C.-H., & Wu, Y.-S. (2018). Hypervisor-Based Sensitive Data Leakage Detector. 2018 IEEE International Conference on Software Quality, Reliability and Security (QRS). pp. 155-162. doi:10.1109/qrs.2018.00029

[4] Tuan, L.H., Cam, N.T. & Pham, VH. Enhancing the accuracy of static analysis for detecting sensitive data leakage in Android by using dynamic analysis. Cluster Comput 22, 1079–1085 (2019). https://doi.org/10.1007/s10586-017-1364-8

[5] G. Kul, S. Upadhyaya and V. Chandola, "Detecting Data Leakage from Databases on Android Apps with Concept Drift," 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), 2018, pp. 905-913, doi: 10.1109/TrustCom/BigDataSE.2018.00129.

[6] R. Baeza-Yates, B. Ribeiro-Neto. Modern information retrieval: The Concepts and Technology behind Search (2nd Edition). ACM Press Books, 2011. pp. 53-55.

[7] C. T. Meadow, R. B.Bert, H. K.Donald. TextInformationRetrievalSystems, 2017. Print.

[8] S. Dominich. Connectionist interaction information retrieval. In: Information processing & management. 2003. pp. 167-193.

[9] Welcome to Stem! URL: https://stem.torproject.org/