Watch out! Doxware on the way...

Routa Moussaileb, Renzo Efrain Navas, Nora Cuppens

Watch Out! Doxware on The Way...

Routa Moussaileb[a,b,*], Renzo E. Navas[a] and Nora Cuppens[c]

[a] IMT Atlantique, Rennes, France
[b] LHS-PEC, Inria, Rennes, France
[c] École Polytechnique of Montréal, Canada

ARTICLE INFO

Keywords:
Doxware
Asset
Exfiltration
Content Analysis
TF-IDF
NLP
Honeypots

ABSTRACT

Malware remains the number one threat for individuals, enterprises, and governments. Malware's aftermath can cause irreversible casualties if the requirements of the attackers are not met in time. Security researchers' primary objective is protecting the assets that a person/company possesses. They are in a constant battle in this cyberwar facing attackers' malicious intent. To compete in this arms race against security breaches, we propose an insight into plausible attacks, especially Doxware (also called leakware). We present a quantification model that explores the Windows file system in search of valuable data. It is based on the Term Frequency-Inverse Document Frequency (TF-IDF) solution provided in the literature for information retrieval. The highest-ranked files will be then exfiltrated over the Internet to the attacker's server. Then, we studied possible countermeasures including deception-based techniques. Amongst the existent ones, we implemented and tested one based on honeypot files and folders to protect users' assets. We conclude by presenting future perspectives in this area with the possible counter-countermeasures that can be used by an attacker to bypass current detection mechanisms. Our approach delivers an observation of the evolution of malware throughout the last years. It enables users to prevent their sensitive information from being exposed to potential risks.

1. Introduction

A Pact with the Devil is always made when a virus executes its payload on the victim's computer as Bond *et al.* state:"The arms race between propagation and defense will continue ad infinitum" [12].

Putting computer security on sounder footing, researchers seek to decrease attacks on companies and end-users. Startling news is conveyed in Symantec's latest report published in 2019 [46]. Even though cryptojacking is down, but not out, targeted attacks blossomed by 78% in 2018. Cloud security and formjacking remain a concern for companies.

Cyber Security kill chain model consists of the attack's structure progressing through several phases. It begins with a reconnaissance and, once the control over the victim's machine is acquired, the payload is executed. This payload marks the objectives of cybercriminals.

Several strategies exist to respond to those cyber-attacks. A well-known malware is ransomware, a type of software that encrypts users' documents asking for a ransom in exchange for the key used for encryption. One countermeasure is the calculation of Shannon's entropy of user's files [28, 29, 40]. In fact, if they are encrypted, their value fluctuates around 8. However, this is a reactive solution. Our goal is to be a step ahead of the attacker to prevent security breaches. Thus, it will give us a better understanding of the possible intrusions.

Analysts also joined the uphill battle against cyber-attacks. It is not affecting end users only; governmental concern is on the rise since it compromises the security and serenity of a country. The ultimate goal of any company is protecting its resources: the data. Data is the most valuable asset a person could acquire. Indeed, it has and is being used for many purposes by the attacker: lucrative opportunities enabling them a monetary gain, for example, blackmailing victims in displaying their private pictures to the public. Company-wise, it could be selling the information gathered to a concurrent one, which will lead to millions of dollars in terms of losses.

Risk evaluation is a necessity in all cases. Companies should take into account the potential danger of disgruntled employees that can jeopardize their supreme interests. Initial leakware threat emerged in the late 2015 with Chimera's ransomware [15]. However, no evidence proves the exfiltration of any personal information. To the best of our knowledge, no previous research was done on a plausible Doxware attack and its feasibility. For all the reasons mentioned above, we endeavor to present Doxware techniques that could be used for victim assets' extortion.

This work is an extension of the conference paper [35], the main new contributions are the countermeasures explored in Section 6 and the discussion Section 7. More particularly:

- We proposed honeypot-based countermeasures (decoy folders and files).
- We implemented and evaluated those countermeasures against our practical doxware attack.
- We offered some insights on possible counter-countermeasures.

Outline The paper is structured as follows. The context and language processing are presented in Section 2. State

Email addresses: routa.moussaileb@imt-atlantique.fr (R. Moussaileb); ruta.moussaileb@irisa.fr (R. Moussaileb); renzo.navas@imt-atlantique.fr (R.E. Navas); nora.boulahia-cuppens@polymtl.ca (N. Cuppens)
ORCID(s): 0000-0002-1938-9656 (R. Moussaileb); 0000-0002-8784-7444 (R.E. Navas)

of the art of data and Natural Language Processing (NLP) is described in Section 3. Our proof of concept (POC) is developed in Section 4. Protection mechanisms are provided in Section 5. We propose, implement, and evaluate a honeypot-based countermeasure in Section 6 and offer some discussion in Section 7. Finally, the conclusion is drawn in Section 8.

2. Context

2.1. From Ransomware To Doxware

Ransomware is a specific type of malware that encrypts victims' files [34]. A second type is ransom-locker that blocks the access to the desktop without the encryption process. Data retrieval can be possible if the required ransom by the attacker is paid. Our primary concern in this paper is crypto-ransomware since they present a higher threat than locker ransomware.

Our proposed approach focuses on the file evaluation for score computation. Doxware samples are not found on public repositories (VirusShare, MalwareDB) or blogs. Therefore, dynamic and static analysis can not be carried out. However, a recent article on Bleeping Computer presents a malware that steals confidential military, and financial files [3]. It is somehow related to ransomware. The sample performs a lookup on all PDF and XLS files, especially those containing words like "fraud", "hack", "tank", "defence", "military", "checking", "classified", "secret", "clandestine", "undercover". Any file that matches those strings is then uploaded via FTP to a remote server. We decided to explore the feasibility of such an attack in a basic user environment and provide ways to delay or stop the executable if possible.

Figure 1 presents ransomware and Doxware workflow. Four main stages appear in both malware (phases P1, P2, P3, and P4). The only difference resides in "valuable files hunting" followed by an exfiltration of the acquired data (phases D-P3 and D-P4). In fact, ransomware attack scope and losses are confined in a users setting: the data remains on the pc, yet, it is encrypted. With a previous backup, end-users can restore all their files. Nonetheless, in a Doxware attack, the damage is beyond repair since once the information is out to the digital world, any person has access to it [27]. To the best of our knowledge, no previous studies were made on this specific type of malware, and it was only mentioned by researches as an advanced threat [18, 33, 36, 42].

Similarly to ransomware, Doxware's attack vectors are mainly phishing/spam emails or unpatched security vulnerabilities on a visited website or on the victim's system [4, 55]. Then, once it is infiltrated on the system, it checks if the required libraries with the appropriate versions are installed on the computer to perform its destructive intents. Afterwards, an exploration of the file system is made to search for example, for military and financial files like in [3]. Specific file types are encrypted (texts, images, and documents) using AES-256 or RSA-2048. Finally, a ransom is demanded to receive the decryption keys and to avoid sensitive information exposure. Besides, some malware authors create eBay-like auction site for stolen data [5].

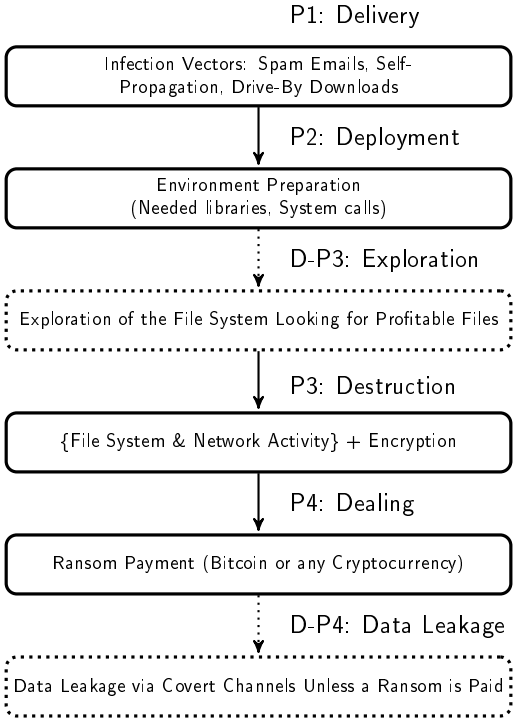

Figure 1: Ransomware (represented by solid lines) Vs Doxware (represented by solid and dashed lines).

2.2. Data Formats Choice

Different data formats exist nowadays that are stored in a computer. They can be classified into three main categories:

1. Textual Documents: They represent files that contain mostly data in the form of a sequence of words or alphabetic characters. For example, contracts, agreements, company's balance sheet, medical records.
2. Pictures: Designs or representations made by various means (such as painting, drawing, or photography). For instance, a Magnetic Resonance Imaging (MRI), gradient descent convergence, trip pictures.
3. Videos: A recording of a motion picture or television program for playing through a television set (movies, video clip, news). These have to be personal in order to blackmail the victim in paying the ransom.
4. Some files have the combination of two or all the categories of previously mentioned types such as a PDF file (it contains paragraphs as well as images).

Nearly all processing methods in the literature for face recognition or body detection are based on machine learning algorithms [11, 45]. Some additional information is mandatory to be able to recognize bodies, clothes, poses.

Many drawbacks reside in these approaches, such as their weight: complex algorithms requiring considerable computation power. This means many false positives cannot be tolerated. Sending a 50 Mb video that does not encompass sensitive information represents a huge loss for the attacker. For example, many packets will be transmitted over the network and cannot go unnoticed. Therefore, a compromise between efficiency and stealthiness is needed. Moreover, pictures of people often represent a red line since you are affecting their privacy, that means they will feel threatened. For all the reasons cited above, our proof of concept developed in this paper is based on textual document analysis, specifically contracts.

2.3. Natural Language Processing (NLP) and Information Retrieval (IR)

The field of NLP encompasses many topics that help to convert, to some extent, a text into a data structure. Statistics, probability, and machine learning were previously used to analyze textual documents. Recently, deep learning techniques are adopted to extract even more refined results in a shorter time due to advances in computational power and parallelization [38]. NLP is adopted in various domains. Hence, a medical NLP is used by Friedman *et al.* to extract molecular pathways from journal articles [21]. Stenetorp *et al.* rely on NLP to develop a Web-based tool for text annotation [43]. Another example worth mentioning is the application of the NLP on hate speech detection presented by Schmidt *et al.* in [41] or Al-Hassan *et al.* in [8].

Information retrieval is defined in an academic field of study as follows: "Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)" [14]. NLP techniques have been used in IR, however, since only few improvements are noticed [13], we focus on presenting the following two state-of-the art techniques used for IR.

- Bag-Of-Words (BOW) is a simple methods used for object categorization. The idea relies on regrouping words by their occurrences.
  Nevertheless, this technique has some well-known flaws. Some words, existent in any kind of documents ("the", "an", "always", "being") called stop words, are not representative of the document itself compared to others. Their frequency might exceed relevant elements in a document. Hence, this technique is backed up by the TF-IDF transformation addressing the problem encountered in BOW [48, 53].
- TF-IDF has the Bag-Of-Words as a basis but with an improved layer. A corpus of files is needed because the process compares documents one to another. Better specificities of the documents can be extracted if there are many specimens as a baseline.
  - TF represents the raw count (frequency) of a given term t in a document d.
    $$tf(t,d)=f_{t,d}$$
  - IDF represents the value carried by the word. IDF score is the logarithm of the number of documents divided by the number of documents that contain the word t. When the number of times a word is present in a document is significant, the value obtained in the logarithm is very close to 1, so idf_t is close to 0. The $idf_{t,D}$ coefficient highlights rare words found only in few documents. Even though not frequent enough, they are significant and are spotted by having a higher score [39, 49].
    $$idf(t,D) = \log\left(\frac{|D|}{|\{d \in D : t \in d\}|}\right)$$
    where:
    $|D|$ is the total number of the documents in the corpus
  - TF-IDF is used as a weighting factor to reflect the importance of a word in a given document or corpus. It is calculated as
    $$tfidf(t,d,D) = tf_{t,d} \cdot idf_{t,D}$$

Latent Dirichlet Allocation (LDA) is a generative probabilistic model of a corpus where each document can be described by a distribution of topics, and each topic can be described by a distribution of words [26]. The algorithm builds the topics from the content of a given corpus. At the end of the execution, the probabilities of the association to the various topics [1...k] are given for each document. LDA uses Bayesian variables to determine those probabilities. LDA is not used in this current proof-of-concept, however, it is kept as future work to tailor the experimental part by applying multiple methodologies for topic representation and comparing the outputs.

3. From Data Encryption to Data Exfiltration

3.1. Where to Find Sensitive Data and How to Track It?

Google Scholar provides more than 5 million research papers regarding sensitive data. It is not limited to a particular field but represents a common concern for a myriad of sectors (healthcare, telecom, automotive, energy). For example, mental health care is a delicate subject that could ruin a person's reputation under malicious manipulation. Therefore, attackers tend to target personal information since it intimidates the victims driving them to pay the ransom in exchange of keeping the information private. Netherlands data breach proves previous hypothesis since it came mostly from the medical sector (29%) [1].

Sensitive information depends on the equipment being used. For instance, Yang *et al.* in [52] consider that the following items represent significant data on Android OS: Unique Device ID, Location, Phone number, Contact book, SMS messages, and Calendar. These items carry

a considerable advantage since each cell phone possesses them, and any application could access them via simple API calls. Taint analysis detects flows upcoming from known and predefined sources (for instance, IMEI of a cellphone) to untrusted sinks like the Internet [9]. Tracking data is, therefore, a straightforward process in Android devices having the predefined sources previously known and the only untrusted sink is the Internet. Taint analysis applied to applications that behold the identified sensitive information helps limiting data leakage. TaintDroid framework developed by Enck *et al.* allows users to monitor how third-party smartphone applications handle their private data in realtime [17]. It uses dynamic taint analysis to track the propagation of tainted data at different levels: instruction level, message-level between applications, and file-level. Original data can be transformed, for example, by writing its content in a pixel bitmap. However, TaintDroid issues warning reports if tainted data is leaked by an application, even if it was transformed since the taint is propagated through these side channels [25]. A similar tool developed by Sun *et al.* enables a multilevel information flow tracking by utilizing registers for taint storage, having only a 15% overhead on the CPU [44]. It presents an enhancement of TaintDroid formerly developed in terms of taint storage and resource consumption [17]. Considerable research is being conducted in this field as in [19, 25, 50, 51].

Data's value is translated by the measure taken by a company to protect it. For instance, Zhu *et al.* provide TaintEraser, a new tool that tracks sensitive user data as it flows through applications [54]. They are one of the pioneers in developing data protection from leakage on Windows OS. Their taint propagation is based on instruction and function level. They evaluate their solution on Notepad, Yahoo!Messenger and the Internet Explorer where they presented accurate results based on taint propagation. However, TaintEraser can be bypassed via data transfer in shared memory. Loginova *et al.* suggest to use cryptographic software to carry out on-the-fly encryption [32]. They state that it represents the most effective approach to overcome data leakage and to protect the information.

On Android OS, attackers know what they are looking for and where to find it, like extracting the victim's GPS location. Yet, these sensitive elements cannot be predefined on a computer level. Indeed, confidential data is only relevant to a particular end-user. For instance, it could be a project for a student or a painting for an artist. Data exists in a variety of formats and are stored in different locations for each user. We analyze in the following parts to which extent data localization is possible on a computer and if it is comparable to mobile devices, and how is the exfiltration carried out.

3.2. Data exfiltration

Data exfiltration is a security breach where this information is disclosed and can be published via the attacker's will. Researchers have long been interested in this domain since it can threaten a company or individual's wellbeing. Giani *et al.* revealed that the bandwidth constraints depend not only on the amount of data exchanged but also on the media being used [23]. Indeed, since 2006 little has changed. Leakage methods remain the same (FTP, SSH, email...).

Al-Bataineh *et al.* presented the detection of malicious data exfiltration in web traffic [7]. Their solution is based on analyzing initially the content of an HTTP POST request (using Shannon entropy) to check whether it is encrypted or not. Additional features were extracted to perform machine learning on the data gathered for malware classification.

Ahmed *et al.* tuned and trained a machine learning algorithm to detect anomalies in DNS queries [6]. Numerous elements are considered like Total count of characters in Fully Qualified Domain Name (FQDN), count of uppercase characters, count of characters in sub-domains, entropy,... Less than 5% of the false positive rate is achieved in their work. Another example is based on Liu *et al.* work, where they were able to detect data theft by analyzing the content of the data being sent to generate a signature [31]. They extracted the information from videos via wavelets enabling them to identify covert communication using Hausdorff Distance.

4. Content Analysis Proposal

It is possible to extract multiple keywords representing a document based on the previous observations and the literature techniques discussed in section 2.3. We make use of this information to check to which extent sensitive data can be disclosed by applying straightforward IR techniques.

4.1. Target Threat Model

The goal of the adversary is to extract sensitive data from a system represented by a personal computer in this analysis. The attacker develops a malicious application that is executed on this system, and once the required information is identified, it is sent through the network to third-party server controlled by the adversary.

We assume that the user installs the malicious application on his computer without prior knowledge of its functionalities or intents. Also, we assume that no counter-measures are in place to protect user's private data.

We focus on the evaluation algorithm that helps to determine and/or to classify data as sensitive or not. The proposed evaluation algorithm depends on some parameters of the document that the program is analyzing. It is divided into two parts.

- **Lexical Generation.** The first part is related to the attacker characterized by generating intelligent lexicons to focus on the subjects/topics he/she wants to exfiltrate. This is accomplished by relying on the TF-IDF methodology applied on a corpus of documents.
- The second part concerns the victim side, where the attacker evaluates victims documents to send them over the network.

- **Document Content Evaluation.** It includes assigning a score to each document based on the keywords or lexicon generated previously.
- **Metadata Evaluation.** In addition, metadata is analyzed to extract further information about the evaluated files.

To sum up, the analysis program consists of 3 modules to accomplish these tasks: **Lexical Generation**, **Document Content Evaluation**, and **Metadata Evaluation**. They are thoroughly explained in the next sections.

4.2. Chosen Corpus: Contracts

Leakware or Doxware is a vast subject and can be interpreted in many ways, whether it infects a personal or a professional machine, an individual user, or a company. Our preference from among choices is the evaluation of professional documents since they can be found on both machines. Textual documents can be saved in various types of extensions (.txt, .docx, .pdf, .rtf , .wpd, .odt...).
Existent tools for word extraction are not applicable on a PDF file containing a scanned document. A possible solution is evaluating them as images, and extracting their content with the help of an Optical Character Recognition (OCR) and Tesseract (an open-source OCR engine). Each page of the PDF document is converted into a .png image. The ratio of PDF to image can be 1/30, a memory consuming process. The program gets importantly less stealthy having a longer processing time compared to .txt. Therefore, we restrain the study domain to .txt, .docx extensions, documents being one of the most targeted files [24].

To carry out the IR process, a corpus of textual documents is required. Contracts are chosen as a topic to represent the sensitive information that is exfiltrated by the attacker. To do so, various files are downloaded from *onecle.com* and *contractsfinder.service.gov.uk*. Three types of contracts are chosen for the corpus:

- investment agreements
- marketing agreements
- partnership agreements

The terms "investment", "marketing", and "partnership" are explicitly marked in the documents. Additional documents are gathered from Google Scholar, online courses, and forums.

The same procedure can be carried out on other topics that convey as well critical information like medical records, biometric data, and political opinions. Thus, the database that contains the pairs of keywords and topics can be extended.

4.3. Evaluation Algorithm

The evaluation algorithm built depends on some parameters of the document that the program is analyzing. It is divided into two parts: one related to the attacker, as in generating intelligent lexicons in order to focus on the subjects/topics she wants to exfiltrate, and another on the victim side, evaluating the documents of the victim to send them over the network. Four elements are required in the analysis program to accomplish these tasks: Lexical Generation, Document Content Evaluation, Password files Evaluation, and Meta Data Evaluation.

4.3.1. Lexical Generation

On the attacker's machine, a pre-processing is made for lexicons generation of any requested subject. The topic represents the files that the attacker is searching for on the victim's machine.

Initially, TF-IDF transformation is applied to the union of documents in the corpus. The top -n (n is a given integer that represents the wanted number of significant words) results represent the words having the highest score for each document.
Table 1 and Table 2 present the ten (n=10) words having the highest TF-IDF score.

The next step relies on creating a function that associates each word in the lexicon to an importance "score". Let w_i be a word that represents the target subject. Let n be the number of words taken into consideration by a document (top n words with highest TF-IDF score). The word w_i has a $p_{i,j}$ position in the document j, the corpus contains N documents. Its value is built as following.

$$Sc(w_i, j) = \frac{n - p_{i,j}}{n}$$

As a result, the total score Sc_i is:

$$Sc_i = \frac{1}{N} \cdot \sum_{j=1}^{n} Sc(w_i, j) = \frac{1}{N} \cdot \sum_{j=1}^{n} \frac{n - p_{i,j}}{n}$$

Sc_i is divided by n for normalization purposes so that any word can have a maximum score of 1. For example, the word "agreement" is the 3^{d} keyword having the highest TF-IDF score in the first document, however, in the second document it is ranked as the 4th most important word.

$$Sc(agreement, Doc.1) = \frac{10 - 2}{10} = 0.8$$

$$Sc(agreement, Doc.2) = \frac{10 - 3}{10} = 0.7$$

The final score is divided by the number of documents considered to maintain the values between 0 and 1.

$$Sc(agreement, Docs) = \frac{0.8 + 0.7}{2} = 0.75$$

A part of the investment agreement lexicon produced for those two documents is presented in Table 3.

At the end of this step, three lexicons are generated, each representing "investment", "marketing", or "partnership" agreements. It is created by applying the total score on the N documents of the chosen corpus.

Word	TF-IDF Score for Doc.1
company	0.3980
section	0.3771
agreement	0.3324
1look	0.2689
purchaser	0.240
shares	0.2290
shall	0.2220
date	0.1773
material	0.1550
closing	0.1508

Table 1
TF-IDF Scores for Document 1.

Word	TF-IDF Score for Doc.2
company	0.5583
buyer	0.4851
shall	0.2415
agreement	0.2166
section	0.2137
acquisition	0.1297
stock	0.1274
securities	0.1106
voting	0.1100
proposal	0.1094

Table 2
TF-IDF Scores for Document 2.

Word	Score
company	1
section	0.75
agreement	0.75
1look	0.35
purchaser	0.3
shares	0.25
shall	0.6
acquisition	0.25
buyer	0.45
stock	0.2
securities	0.15

Table 3
Score of Words.

Word	Number of Occurrences	Lexicon Score
party	1	0
promises	1	0
commence	1	0
relevant	1	0
applications	1	0
the	3	0
following	1	0
execution	1	0
agreement	1	0.75

Table 4
Content Evaluation.

Statistics	Random Non Decoy Files	Important Files
Min	0.503	1.31
Max	5.66	7.698
Avg	1.653	4.29
σ	2.045	1.359

Table 5
Statistics of the Scores of Random and Important Files.

4.3.2. Document Content Evaluation

The lexicons are already embedded in the malware source code on the victim's side. They are used to process a content score which is combined with a metadata score for a final evaluation score.

At first, a dictionary containing every word of the document with its number of occurrences is extracted. The initial value of the content score is 0.
Let CS be this content score, Sc_i the score of the word i of the lexicon being studied created previously, n the number of words in the lexicon and occ_i the number of occurrences of the word i in the document analyzed.

$$CS = \sum_{i=1}^{n} Sc_i \cdot occ_i$$

Let us consider a document composed of the following sentence: "Party B promises to commence the relevant applications under the Project within 15 days following the execution of this agreement." The content evaluation is presented in Table 4 and in the following equation.

$$\begin{aligned} CS &= (1 \cdot 0) \cdot 7 + 3 \cdot 0 + 1 \cdot 0.75 \\ &= 0.75 \end{aligned}$$

The content score is calculated for each of the three previously generated lexicons. The highest score is retained, followed by a division by the size of the document. This division helps to maintain the same probability to have important documents regardless of the size if they carry the same information.

$$CS_{\text{final}} = \frac{1}{document\_size} \cdot \max\left(CS_1, CS_2, CS_3\right)$$

where 1, 2, and 3 are respectively "investment", "marketing", and "partnership". We select randomly 36 documents that do not contain any information about contracts such as medical records, statistics, state of the art of ransomware, etc. and we compare the scores to the 36 contracts in our corpus. The minimum (Min), the maximum (Max), the average (Avg), and the standard deviation (σ) of the obtained scores are presented in Table 8.

4.3.3. Metadata Evaluation

Metadata is "a set of data that describes and gives information about other data". This information is stored in various file types and its analysis describes the considered document.

Fifteen metadata can be accessed and extracted from a Word document. Those are author, category, comments, content status, created, identifier, keywords, language, last modified by, last printed, modified, revision, subject, title, and version. These details are useful for an attacker to determine if the analyzed file is important to the user. It is accomplished by checking the date of creation and modification. In fact, old files are of no interest for exfiltration whereas recent files used by end-users carry relevant information. In addition, the multiple revision of a file indicates that the user is manipulating this document, therefore, it contains pertinent information.

Most of the cited metadata analyzed are not filled in (the content is null). Some metadata have to be manually annotated like keywords to support searching and indexing, as well as the comments part to describe the content of the resource. Therefore, the only features kept and the most relevant ones are: the number of revisions, created, and last printed.

- If more than 1 revision is made, the metadata score is incremented by 5.
- If the document is created in 2019 (the timeline where the experiments were made), the metadata score is incremented by 1.
- Additionally, if the document is printed in 2019, the metadata score is incremented by 2.

The algorithm Valuable File Hunting (VFHA) summarizes the steps developed in our paper.

4.4. Proposal Summary

This section summarizes the previous steps taken to achieve a complete scan of the file system of the victims computer in search if valuable files, more specifically, contracts. It is portrayed by the algorithm Valuable File Hunting (VFHA).

1. Initially, the lexicon are generated based on the contract topic (line 2 in VFHA).
2. Then, the parsing of the target file system is made searchings for .txt and .docx extensions (line 9 in VFHA).
3. The file score is calculated as following.
   (a) The content of .txt file is extracted, and vocabulary analyzed. Each word is compared with a lexicon previously created that contains recurrent and relevant words in a contract based document.
   (b) The same procedure is done for the .docx files. However, an additional step is made for the metadata analysis. The significant metadata are the number of revisions, creation date, and the date when it was last printed. They are added to the total sum representing the value of a document (line 3 and 9 in VFHA).

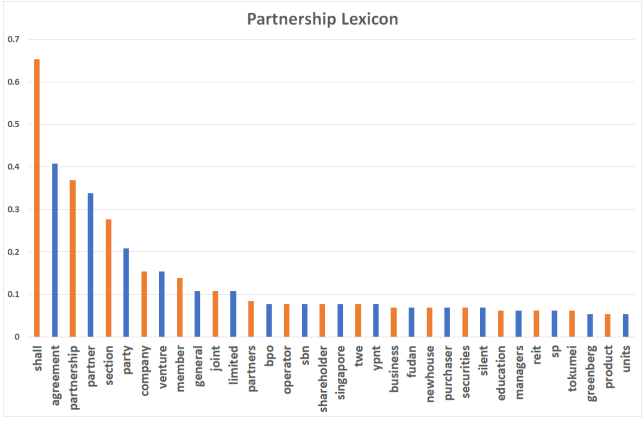

Figure 2: An Example of Lexicon with the associated Scores.

4. "Summarize" step: Each document has a total score that has been assigned, so the list of tuples (path, score) is sorted according to the value obtained, where the attacker chooses which ones he/she wants to extract. For instance the first 50 files (line 23 and 24 in VFHA).

Algorithm Valuable File Hunting VFH

```
1:  procedure ALGORITHM 1
2:      Topic_Lexicon ← {lexicon_generator (Contracts) }
3:      def analyse_content(File f):
4:      for word ∈ f do
5:          if word ∈ Topic_Lexicon then
6:              f_score += score(word) * number_occurrences
7:      return f_score/len(f)
8:
9:      def analyse_metadata(File f):
10:     if f.core_properties.revision > 1 then
11:         f_metadata_score += 5
12:     else if f.created == "2019" then
13:         f_metadata_score += 1
14:     else if f.lastprinted == "2019" then
15:         f_metadata_score += 2
16:     return (f_metadata_score)
17:
18:     Parse the File System
19:     if FileExtension ∈ .txt or .docx then
20:         FileList ← {Analyse MetaData and Content}
21:     else
22:         Continue;
23:     Sort FileList by highest_Score
24:     Send n first valuable files to the attacker's server (future work)
```

4.5. Lexicon Generation

The chart presented in figure 2 shows the most common words in a contract document are: "shall", "partnership", "agreement" and also "section". Indeed, the corpus gathered

Document	Keywords
Storm.docx	"section", "application", "online", and "programs"
Food-processing.docx	"company", "material", "date", and "investment"
TF-IDF.docx	"section", "documents", "management", and "limited"
Csquotes.docx	"section", "closing", "boundary", and "units"
Localization-bieber.docx	"company", "article", "series", and "advance"

Table 6
Retrieved Keywords from the Noise Documents.

is previously identified as a contract type document, which is an advantage allowing us to perform a relatively simple algorithm to determine whether a document belongs to this category or not, although law documents also acquire an important score and may also be as valuable as a contract.

4.6. Valuable Files Chase Results & Limitations

A Windows 7 Virtual Machine is created for the proof of concept. It holds 50 noise documents, and 10 contracts are added. After running the algorithm, 15 files are selected. Among those, there were seven false positives, where 5 are the noise files and 2 are Windows configuration files.

For example, a Python "README.txt" note is the following:
"*This directory exists so that 3rd **party** packages can be installed here. Read the source for site.py for more details.*"

It contains a single occurrence of the keyword **party**. However, the final score depends on the file size. Since it is only 121 bytes, relatively small compared to other texts on the file system, it is associated with a high score.

Besides, the "formatfloat_testcases.txt" of the Python library contains multiple sequences of numbers that infer the presence of a password ("123456"). However, the document solely explains the formatting instructions in practice that Python uses.

The noise documents are the following:

- Storm.docx represents a documentation of Storm a free and open source distributed real-time computation system.
- Food-processing.docx is a review of the efforts and inventions in field of emerging food processing technologies since their inception to present day.
- TF-IDF.docx is an article explaining the TF-IDF formula and its advantages.
- Csquotes.docx provides explanation about the csquotes package used in Latex.
- Localization-bieber.docx represents the special string used in Latex.

Specific Windows file system path can be removed from the file system traversal, in addition to "README.txt" related documents to gain better results. A limitation of extracting keywords from documents using solely TF-IDF is the absence of the semantic context. As seen previously, a "README" document or documentation can contain some specific keywords like "party" but not relevant to contracts. Table 6 shows some selected keywords contained in the noise documents. Even though different contexts and semantics are considered (food processing or latex explanation), they still carry some information related to contracts. Therefore, the use of the sequence of n-keywords for a given topic provides an accurate context of the document being analyzed. This reduces false positives.
Besides, since each word is considered independently, multiple scores can be attributed to the same root of the keyword. For example, "parties" score is 0.1062 whereas "party" score is 0.2804. It is convenient to reduce inflected (or derived) words to their root form and remove inflectional endings to assign a unique score to the root of the word. Those two concepts are defined as Stemming and Lemmatization [47, 10, 37].

5. Security Recommendations

Protection against malware attacks, especially zero-days, is a challenge for all researchers. Residual risk remains: de facto, despite various countermeasures employed by a party, an attacker can always find a way to penetrate the system (he/she still risks to be detected). If committed, anyone can reach their malicious intent. However, our goal is to complicate the intrusion task, detect it if possible, rather than handling it to the attackers on a golden plate. Users should know the existent vulnerabilities to see what patches can be used to circumvent malevolent attacks. Some countermeasure can be deployed by users to protect their data from being exfiltrated:

1. Honeypot Folders: They can be created in any environment, regardless of the operating system used. Since doxware will traverse the whole file system looking for assets, any process or thread that will pass through this lure folder can be immediately flagged then stopped. A drawback would be malware's multithreading techniques, it can still be exposed but after a certain epsilon time.
2. Data Tainting: Sensitive data in a computer is extremely private and depends on the end users, unlike

Android OS (IMEI, GPS location,... existent on all mobiles). Therefore, a general protection model is impossible to develop in real life. However, each individual can add a layer, a taint, on his preferred/sensitive information. Thus, each exfiltration attempt over the network will be detected. Nonetheless, a person can have an explosion of tainted data that may slow down the system.

3. Data Encryption: It remains a robust way adopted by the global community. Indeed, brute-forcing the encryption key can take decades. Even though an attacker acquired the encrypted files, he/she cannot menace the victims or blackmail them since no access to the decrypted data is possible.

6. Honeypot in the case of a Doxware attack

In this section, we propose honeypot-based countermeasures against Doxware attacks. First, we present recent work done in the literature to stop ransomware using honeypot techniques. Then, we propose a combination of honeypot folders and files to detect and mitigate file exfiltration (Doxware) attacks. Finally, we implement and evaluate our proposal.

6.1. Honeypot Key Elements

- An essential aspect to take into consideration while generating decoys is the location of the honeypot files and folders. After performing a static analysis of 11 ransomware samples, the authors in [30] conclude that they traverse the file system in the following order:
  1. C:/User/USER NAME/Documents
  2. C:/
  3. C:/User/USER NAME/Desktop
  4. C:/User/USER NAME/Favorites

- Since, the traverse can be in order or reverse order of Windows-1252, folders names should be created accordingly to increase the chances of being accessed first. Consequently, attacks can be stopped rapidly after their initial execution. This information is useful since Doxware represents ransomware upgraded version (section 6.2.1).

- Two types of decoy files exist Low and High Interaction. Low interaction decoy files contain arbitrary data to detect ransomware's actions on the documents, whereas high interaction decoy files contain false information to confuse the attacker and lead him to further deception mechanisms implemented on the system [16]. The latter is used in the use case since Doxware tackles the content of files. The generated decoy content relies on lexicons frequency distribution (section 6.2.2).

6.2. Proposal Overview

Our honeypot-based proposal relies on the generation of decoy folders and files. The procedure for decoy folder name generation is described in Section 6.2.1. The process for decoy file content generation is described in Section 6.2.2. We implement and evaluate our proposal, and offer some discussion in Section 6.3.

6.2.1. Decoy Folder Name Generation

As stated in section 6.1, three crucial elements need to be considered while developing and deploying honeypot folders: their name, location, and content.
Applications, including ransomware, carry out file operations by calling two Windows APIs, *FindFirstFile* and *FindNextFile* [20]. Thus, the following decoy name generation methodology is used to be the first folder selected in the file system traversal.

1. All the repositories in a specific path are sorted by ascending order. Then, the first and last folder are chosen in the list.
2. To create the corresponding honeypot folder, the first character in the chosen folder is selected, and represented in hexadecimal. Then, a subtraction or addition is performed to embody the first and the last folders that are traversed by malicious processes in the case of a Doxware attack.
3. The string is appended to a random word extracted from "The Brown Corpus" [2]. The first character allowed by Windows-1052 is chosen for naming repositories that is the space character (0x20) and the last one, which is the letter ž (0x9e).

Figure 3 displays a list of the names of all files in the current working directory "Desktop". The first folder is "2020" and the last one "Work-Documents". They are selected as a baseline for generating decoy folders names.

$$\begin{aligned} FirstDecoyName &= concat(\text{␣}, \text{␣}, (hex(2)-1), randomString) \\ &= concat(\text{␣}, \text{␣}, 1, general) \\ &= \text{␣␣}1general \end{aligned}$$

6.2.2. Decoy File Content Generation

We use Honeypot (or Decoy) Files as a proactive counter-measure against the Data exfiltration attack (i.e., Doxware malware) proposed in this paper. We propose to use Honeypot Files as a complementary measure to the Honeypot Folders detailed in 6.2.1.

Rationale. Before any attack, we generate and store *honeypot files* that semantically resemble relevant, targeted documents. Those files contain no meaningful information. However, an attacker as advanced as the one of Section 4 (i.e., using the VFH Algorithm) will flag those files as valuable and exfiltrate them. Hence, files with real sensitive information will have less probability of being exfiltrated, as the attacker will not prioritize them..

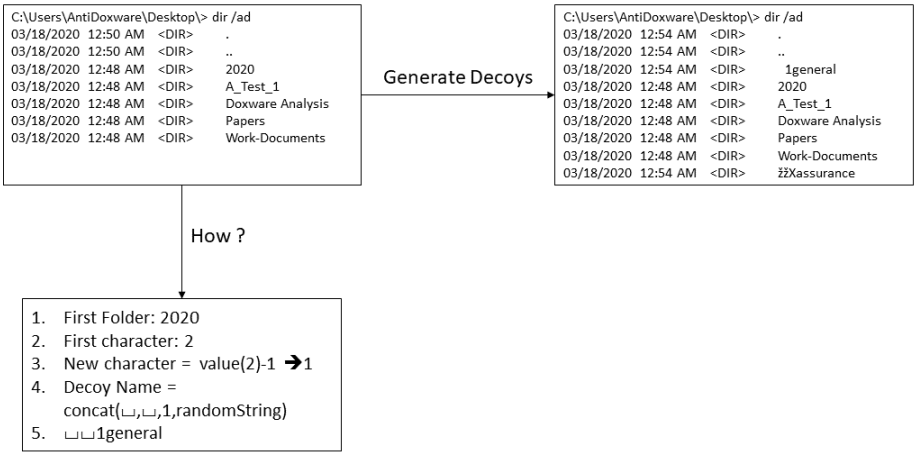

Figure 3: Decoy Folders Generation.

Honeypot Files Generation. Our proposal assumes that we partially know the VFH Algorithm (*Hypotesis*). The file generation is based on the construction of a *lexicon* (Section 4.3.1) that corresponds to the Subject of the files we want to protect. A *lexicon* consist of n tuples ($word_i$, $weight_i$) $i \in \{1, ..., n\}$. A $weight_i$ has a value between 0 and 1, but the sum of all weights is not 1. We propose to normalize them so this sum equals 1. This normalized weight p_i now can be interpreted as the relative frequency of $word_i$ in the given *lexicon*. With p_i we can construct the probability mass function (pmf) of *word* taken as a discrete random variable W. Finally, a Honeypot File HF consists of a succession of N independent and identically distributed *word* random variables $HF = \{W_1, W_2, ..., W_N\}$.

Note. *HFs* constructed this way do not guarantee the maximum value according to the VFH Algorithm. The maximum value will be obtained by the repetition of the single word with the highest weight. However, *HFs* generated with our proposal (i.e., independent and lexicon-based distributed) will be more diverse and semantically complex, providing a better resemblance to reality than a single-word-repetition approach.

The decoy generation proposal is resumed in Algorithm Decoy Folders Creation (DFC).

Algorithm Decoy Folders Creation DFC

```
1:  procedure ALGORITHM 1
2:      def Create_Decoy(Path path):
3:      folders ← {sorted_alphabetical_order(GetFolders)}
4:      GenerateDecoyNames(folders) # Section 6.2.1
5:      for f ∈ DecoyFolders do # We generate decoy files
        inside the folders
6:          GenerateDecoyFiles() # Section 6.2.2
7:      return DecoyFoldersPath
8:
9:      Monitor Decoy Folders
10:     if changeOccured ( DecoyFolders and / or Decoy-
        Files) then
11:         ALERT!!!
12:     else
13:         Continue;
```

6.3. Implementation and Evaluation

We implemented a proof of concept of our proposal in Python 3 that generates Decoy Folder and Files as specified previously.

Initially, we generate 500 decoy files of different length containing from 50 to 8000 words. We calculate the minimum, the maximum, the average and the standard deviation as seen in Table 7.

Regardless of the size, consequently, the number of words generated, the score of a specific file fluctuates in general around the value 23. If the number of the words per file increases, the minimum value will increase too since the document contains more and more essential lexicons that represent an important file. The maximum value decreases since it takes into consideration the size of the file.

Then we selected randomly 36 documents that do not contain any information about contracts such as medical records, statistics, state of the art of ransomware, etc. and we compared the scores to the 36 contracts in our corpus (Table 8).

Statistics	50-Words	200-Words	500-Words	1000-words	2000-Words	4000-Words	8000-Words
Min	13.316	15.734	16.715	17.438	17.787	17.9	18.259
Max	36.626	32.336	31.465	30.106	29.482	29.259	29.026
Avg	23.692	23.357	23.071	23.031	23.035	22.979	22.999
σ	4.717	4.133	3.967	3.963	3.935	3.892	3.888

Table 7
Statistics of the Scores of Different Decoy Files.

Statistics	Random Non Decoy Files	Important Files
Min	0.503	1.31
Max	5.66	7.698
Avg	1.653	4.29
Std-Deviation	2.045	1.359

Table 8
Statistics of the Scores of Random and Important Files.

7. Discussion

In this work, we provide a proof-of-concept Doxware attack based on information retrieval. Our proposal selects a limited number of documents, but of high value. Unlike other *brute-force* attacks, like classical ransomware, this low-quantity-high-quality approach allows for a stealthy exfiltration phase. This makes Doxware attacks more difficult to detect and mitigate.

Countermeasures. In Section 5, we present three families of possible counter-measures against exfiltration attacks. We propose a honeypot-based countermeasure in Section 6 using both honeypot (decoy) folders and files. It detects and mitigates file exfiltration (Doxware) attacks. The combination of having distributed decoy folders in different paths and a rich, readable content provides protection in two levels: the first one helps to flag any suspicious process passing through those decoys (it is not restricted to Doxware attacks). The content helps to mislead the attacker into believing that he/she possesses valuable data. If the attacker performs a recursive lookup in an alphabetical order for specific files, he/she will be instantly blocked since he passed through decoys. The watchdog observer will raise an alert immediately. Considering the attacker was able to bypass decoy folders, the data he would have collected revolves mainly about keywords extracted from the contracts lexicon, which does not represent an information of value for any given individual or company.

Counter-countermeasures. Our counter-measure assume that the file selection criteria of the adversary (the Valuable File Hunting Algorithm) is known. If the adversary is aware of this fact, it can, in turn, modify its behavior to mitigate the defensive measures. In the remaining of this section, we discuss some possible adversarial behavior evolution in this direction.

Avoiding decoy folders. An adversary can avoid altogether the n folders that are first and last in alphabetical order traversal. It can also do an iterative Doxware attack, from low to high risk of running into decoy folders. A single iteration will include the exfiltration phase of the files. Thus, even if the adversary is detected and neutralized, the files from previous iterations would have been already exfiltrated. E.g., the first iteration will avoid not only the n folders as mentioned before, but also well-known folder names such as "Documents", "Desktop", or "Favorites".

Avoiding decoy files. File exfiltration and its countermeasure are based on the VFH Algorithm and the particular lexicons. The same or very similar lexicons are assumed to be known by both parties. A possible decoy file avoidance behavior is to use a specially tailored lexicon that targets very specific words. Particularly, this lexicon should avoid the words that have highest scores from the assumed well-known lexicon. Another lexicon-based solution is to use lexicons and an algorithm that weights the frequency not of single words but of tuples or triplets. We can also assign negative weight to semantically incorrect combinations of tuples, that will only occur in our artificially generated single-word-based decoy files. Another type of decoy-file avoiding measure is to apply statistical analysis over the scores of potential files to exfiltrate. We studied the statistical properties of the decoy files of our proposal (See Table 7), e.g.,: average score and standard deviation as a function of number of words. Thus, the adversary can avoid the files that have values similar to the expected mean$\pm\sigma$ of decoy files. An histogram or probability mass function of the scanned files can also alert the adversary of potential decoys: if many scanned files are found to be of high score (e.g., the right tail of the probability mass function concentrates many of the values), it is likely that all of them are decoy files and should be discarded. The exfiltration should be done analyzing only the remaining files.

To conclude this section, we remark that once the defender is aware of this possible adversarial behavior, it can also modify its own in order to remain effective. How much knowledge we have about the other party and how many resources are at our disposal will determine the best strategy to follow.

8. Conclusion

We have discussed in this paper the potential danger of sensitive data localization and quantification that can be carried out by a Doxware attack. Windows OS is the target system throughout the experiments. A proof of concept is developed based on the contract topic that includes investment, marketing, and partnership agreements. State of the art methods were used, such as TF-IDF and Bag of Words, in addition to a document's metadata. The associated score of each document is calculated then normalized. Few samples

of files regarding the same topic are needed to identify new target topics. Even if victims are aware that some of their sensitive information was leaked, there are no measures in place that can mitigate the damage caused by a Doxware. Reducing the false positive rate can be done by eliminating the Windows system path and choosing randomly N last visited files in Windows' Quick Access. Implementing a honeypot-based countermeasure requires an analysis of the targeted system. Multiple key elements should be taken into consideration and finely tuned like the content of the files as well as their location. Once deployed, high and low interaction decoy files can detect the attack process. However, there is no guarantee that the decoy will be traversed at the beginning of the infection process. Distributed repositories should be placed in the file system containing randomly generated files of a predefined content to maximize the chances of succeeding using honeypot methodology. Re-designing decoys generation of the deception-based techniques improves the protection of the data of users, as mentioned in [22]. Threats arising from this cyber warfare are exponential. Therefore, end-users should be aware of the possible attacks, mainly attack vectors, in order to avoid and circumvent them protecting their assets.

Acknowledgments

We would like to thank Charles Berti, Guillaume Deboisdeffre and Jean-Louis Lanet for their contributions.

References

[1] , . Data Protection and Privacy across sectors and borders . https://bit.ly/2D2r77M.

[2] , 2012. Accessing text corpora and lexical resources. URL: https://www.nltk.org/book/ch02.html.

[3] , 2019. Ryuk Related Malware Steals Confidential Military, Financial Files. https://www.bleepingcomputer.com/news/security/ryuk-related-malware-steals-confidential-military-financial-files/?fbclid=IwAR09GDZRRVV7C_QHDzNgf1SqfMWjdGGsfjFhg2pDRJTXp9W9mKhej9cGk-A.

[4] , 2020. What is Doxware? https://www.securemac.com/blog/what-is-doxware.

[5] Abrams, L., 2020. Revil ransomware creates ebay-like auction site for stolen data. https://www.bleepingcomputer.com/news/security/revil-ransomware-creates-ebay-like-auction-site-for-stolen-data/.

[6] Ahmed, J., Gharakheili, H.H., Russell, C., Sivaraman, V., 2019. Real-time detection of dns exfiltration and tunneling from enterprise networks. Proceedings of IFIP/IEEE IM, Washington DC, USA .

[7] Al-Bataineh, A., White, G., 2012. Analysis and detection of malicious data exfiltration in web traffic, in: 2012 7th International Conference on Malicious and Unwanted Software, IEEE. pp. 26–31.

[8] Al-Hassan, A., Al-Dossari, H., 2019. Detection of hate speech in social networks: a survey on multilingual corpus, in: 6th International Conference on Computer Science and Information Technology.

[9] Arzt, S., Rasthofer, S., Fritz, C., Bodden, E., Bartel, A., Klein, J., Le Traon, Y., Octeau, D., McDaniel, P., 2014. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. Acm Sigplan Notices 49, 259–269.

[10] Balakrishnan, V., Lloyd-Yemoh, E., 2014. Stemming and lemmatization: a comparison of retrieval performances .

[11] Bartlett, M.S., Littlewort, G., Frank, M., Lainscsek, C., Fasel, I., Movellan, J., 2005. Recognizing facial expression: machine learning and application to spontaneous behavior, in: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), IEEE. pp. 568–573.

[12] Bond, M., Danezis, G., 2006. A pact with the devil, in: Proceedings of the 2006 workshop on New security paradigms, ACM. pp. 77–82.

[13] Brants, T., 2003. Natural language processing in information retrieval, in: CLIN.

[14] Cambridge, U., 2009. Introduction to information retrieval .

[15] Constantin, L., 2015. New ransomware program threatens to publish user files. https://www.computerworld.com/article/3002120/new-ransomware-program-threatens-to-publish-user-files.html.

[16] El-Kosairy, A., Azer, M.A., 2018. Intrusion and ransomware detection system, in: 2018 1st International Conference on Computer Applications & Information Security (ICCAIS), IEEE. pp. 1–7.

[17] Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B.G., Cox, L.P., Jung, J., McDaniel, P., Sheth, A.N., 2014. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. ACM Transactions on Computer Systems (TOCS) 32, 5.

[18] Ensey, C., 2017. Ransomware has evolved, and its name is doxware. DARKReading. nformationWeek Business Technology Network .

[19] Feng, Y., Anand, S., Dillig, I., Aiken, A., 2014. Apposcopy: Semantics-based detection of android malware through static analysis, in: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, ACM. pp. 576–587.

[20] Feng, Y., Liu, C., Liu, B., 2017. Poster: A new approach to detecting ransomware with deception, in: 38th IEEE Symposium on Security and Privacy.

[21] Friedman, C., Kra, P., Yu, H., Krauthammer, M., Rzhetsky, A., 2001. Genies: a natural-language processing system for the extraction of molecular pathways from journal articles, in: ISMB (supplement of bioinformatics), pp. 74–82.

[22] Genç, Z.A., Lenzini, G., Sgandurra, D., 2019. On deception-based protection against cryptographic ransomware, in: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Springer. pp. 219–239.

[23] Giani, A., Berk, V.H., Cybenko, G.V., 2006. Data exfiltration and covert channels, in: Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense V, International Society for Optics and Photonics. p. 620103.

[24] Gonzalez, D., Hayajneh, T., 2017. Detection and prevention of crypto-ransomware, in: 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), IEEE. pp. 472–478.

[25] Graa, M., Cuppens-Boulahia, N., Cuppens, F., Lanet, J.L., Moussaileb, R., 2017. Detection of side channel attacks based on data tainting in android systems, in: IFIP International Conference on ICT Systems Security and Privacy Protection, Springer. pp. 205–218.

[26] Jelodar, H., Wang, Y., Yuan, C., Feng, X., Jiang, X., Li, Y., Zhao, L., 2019. Latent dirichlet allocation (lda) and topic modeling: models, applications, a survey. Multimedia Tools and Applications 78, 15169–15211.

[27] Keshavarzi, M., Ghaffary, H.R., 2020. I2ce3: A dedicated and separated attack chain for ransomware offenses as the most infamous cyber extortion. Computer Science Review 36, 100233.

[28] Kharaz, A., Arshad, S., Mulliner, C., Robertson, W., Kirda, E., 2016. {UNVEIL}: A large-scale, automated approach to detecting ransomware, in: 25th {USENIX} Security Symposium ({USENIX} Security 16), pp. 757–772.

[29] Kharraz, A., Robertson, W., Balzarotti, D., Bilge, L., Kirda, E., 2015. Cutting the gordian knot: A look under the hood of ransomware attacks, in: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Springer. pp. 3–24.

[30] Lee, J., Lee, J., Hong, J., 2017. How to make efficient decoy files for ransomware detection?, in: Proceedings of the International Conference on Research in Adaptive and Convergent Systems, pp. 208–212.

[31] Liu, Y., Corbett, C., Chiang, K., Archibald, R., Mukherjee, B., Ghosal, D., 2008. Detecting sensitive data exfiltration by an insider attack, in: Proceedings of the 4th Annual Workshop on Cyber Se-

curity and Information Intelligence Research: Developing Strategies to Meet the Cyber Security and Information Intelligence Challenges Ahead, ACM, New York, NY, USA. pp. 16:1–16:3. URL: http://doi.acm.org/10.1145/1413140.1413159, doi:10.1145/1413140.1413159.

[32] Loginova, N., Trofimenko, E., Zadereyko, O., Chanyshev, R., 2016. Program-technical aspects of encryption protection of users' data, in: 2016 13th international conference on modern problems of radio engineering, telecommunications and computer science (TCSET), IEEE. pp. 443–445.

[33] Lueders, S., et al., 2017. Computer security: Enter the next level: Doxware .

[34] Luo, X., Liao, Q., 2007. Awareness education as the key to ransomware prevention. Information Systems Security 16, 195–202.

[35] Moussaileb, R., et al., 2020. Watch Out! Doxware on the Way..., in: CRiSIS 2019: Risks and Security of Internet and Systems, Springer International Publishing. pp. 279–292. doi:10.1007/978-3-030-41568-6_18.

[36] Nadir, I., Bakhshi, T., 2018. Contemporary cybercrime: A taxonomy of ransomware threats & mitigation techniques, in: 2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), IEEE. pp. 1–7.

[37] Nicolai, G., Kondrak, G., 2016. Leveraging inflection tables for stemming and lemmatization., in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1138–1147.

[38] Otter, D.W., Medina, J.R., Kalita, J.K., 2020. A survey of the usages of deep learning for natural language processing. IEEE Transactions on Neural Networks and Learning Systems .

[39] Ramos, J., et al., 2003. Using tf-idf to determine word relevance in document queries, in: Proceedings of the first instructional conference on machine learning, Piscataway, NJ. pp. 133–142.

[40] Scaife, N., Carter, H., Traynor, P., Butler, K.R., 2016. Cryptolock (and drop it): stopping ransomware attacks on user data, in: 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), IEEE. pp. 303–312.

[41] Schmidt, A., Wiegand, M., 2017. A survey on hate speech detection using natural language processing, in: Proceedings of the Fifth International workshop on natural language processing for social media, pp. 1–10.

[42] Sherer, J.A., McLellan, M.L., Fedeles, E.R., Sterling, N.L., 2016. Ransonware-practical and legal considerations for confronting the new economic engine of the dark web. Rich. JL & Tech. 23, 1.

[43] Stenetorp, P., Pyysalo, S., Topić, G., Ohta, T., Ananiadou, S., Tsujii, J., 2012. Brat: a web-based tool for nlp-assisted text annotation, in: Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics, pp. 102–107.

[44] Sun, M., Wei, T., Lui, J., 2016. Taintart: A practical multi-level information-flow tracking system for android runtime, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ACM. pp. 331–342.

[45] Sun, X., Wu, P., Hoi, S.C., 2018. Face detection using deep learning: An improved faster rcnn approach. Neurocomputing 299, 42–50.

[46] Symantec, 2019. Internet Security Threat Report by Symantec. https://www.symantec.com/content/dam/symantec/docs/reports/istr-24-executive-summary-en.pdf.

[47] Torres-Moreno, J.M., 2012. Beyond stemming and lemmatization: Ultra-stemming to improve automatic text summarization. arXiv preprint arXiv:1209.3126 .

[48] Wallach, H.M., 2006. Topic modeling: beyond bag-of-words, in: Proceedings of the 23rd international conference on Machine learning, ACM. pp. 977–984.

[49] Wu, H.C., Luk, R.W.P., Wong, K.F., Kwok, K.L., 2008. Interpreting tf-idf term weights as making relevance decisions. ACM Transactions on Information Systems (TOIS) 26, 13.

[50] Xue, L., Qian, C., Zhou, H., Luo, X., Zhou, Y., Shao, Y., Chan, A.T., 2019. Ndroid: Toward tracking information flows across multiple android contexts. IEEE Transactions on Information Forensics and Security 14, 814–828.

[51] Yan, L.K., Yin, H., 2012. Droidscope: Seamlessly reconstructing the {OS} and dalvik semantic views for dynamic android malware analysis, in: Presented as part of the 21st {USENIX} Security Symposium ({USENIX} Security 12), pp. 569–584.

[52] Yang, Z., Yang, M., 2012. Leakminer: Detect information leakage on android with static taint analysis, in: 2012 Third World Congress on Software Engineering, IEEE. pp. 101–104.

[53] Zhang, Y., Jin, R., Zhou, Z.H., 2010. Understanding bag-of-words model: a statistical framework. International Journal of Machine Learning and Cybernetics 1, 43–52.

[54] Zhu, D.Y., Jung, J., Song, D., Kohno, T., Wetherall, D., 2011. Tainteraser: Protecting sensitive data leaks using application-level taint tracking. ACM SIGOPS Operating Systems Review 45, 142–154.

[55] Zimba, A., Simukonda, L., Chishimba, M., 2017. Demystifying ransomware attacks: Reverse engineering and dynamic malware analysis of wannacry for network and information security. Zambia ICT Journal 1, 35–40.