Ransomware: Analysis of Encrypted Files

Houria MADANI¹, Noura OUERDI², Abdelmalek Azizi³ Faculty of sciences, Mohammed First University, Oujda, Morocco^{1, 2, 3}

Abstract—Ransomware is a type of malware that damage the system by encrypting all the files existing in the computer. To get access, the victim has to pay a ransom to get a key to decrypt his data. When the virus is running in machine, the user cannot stop it on the first try, so he may lose his entire files. One of the goals of this work is to detect ransomware based on encrypted files in real time and to minimize the cost of losing files. We will try to do an analysis of a received file (without opening it and seeing its contents). This scanning action can prevent a ransomware from spreading in the system. Most Ransomware files are sent in ".exe" format, but in this work, we will try to use other file formats that can accept malware, for example, .doc or .docx, .xls or .xlsx, .ppt or .pptx, .jpg, etc. In fact, an attacker can focus only on the files that contain useful data. In this paper, we are going to identify the types of files if they are suspicious or normal (without opening them) from their headers. For that first, we are going to analyze each extension separately (.docx, .exe, .pptx, .xlsx, .jpg, etc.) by identifying their headers and signatures. Then we will take several files with different extensions to analyze them by doing a program who detect if a file is benign or suspicious.

Keywords—Ransomware; encrypted files; signature; file format; static analysis

I. INTRODUCTION

In recent years, ransomware attacks continue to explode exponentially around the world; the cost keeps falling and exploit different sectors.

Researchers and cybersecurity specialists are still looking for a solution to detect this attack and even to slow down its growth in order to find an effective and reliable solution. We see many solutions, but not 100% sure, because hackers are always attentive and updated with the new technologies, they use more sophisticated techniques to follow the evolution and bypassing the protection techniques.

This study focuses on the examination of the behavior and method in which ransomware encrypts files. Ransomware can infiltrate a device in various formats like .exe, .docx, .ppt, etc. A user may open a .docx file without realizing it is an unsafe file that contains metadata that can damage their computer. Therefore, we aim to analyze the files (without opening them) before and after ransomware encryption, in order to distinguish between a typical file and a suspicious one.

In this paper, we will make a study on files to differentiate between a normal file and a suspicious one. For that in Section II, we will approach some "state of the art" concerning the study of files to give you an idea of the current research on this subject. In Section III, we will see our objectives and working methodology to identify and detect a normal file from another suspect one. We will discuss the results that we have had in Section IV. At the end, we sum up with a conclusion and some perspectives.

II. STATE OF THE ART

As you know, attackers are very inventive when they want to target a victim and we find, often, that emails are the trickiest (more than 90%) way [1] for them to create a link between the attacker and the target. Fig. 1 explains how ransomware attacks your machine:

Ransomware detection techniques [2]–[5] are becoming more and more competitive, and each researcher has his own method and technique. If we take the detection of ransomware or malware in general, using file headers, several researchers work focus on a single file extension like PE (Portable Executable) files [6]–[8], but there is not enough research on the detection of ransomware using the headers of different extensions.

The authors in [9] proposed a new classification model based on machine learning techniques to detect and classify malicious and benign PE files based on their headers information. The experimental results proved that the Random Forest algorithm yields a higher accuracy (99.68%) compared to other algorithms. The tests were performed on 211,067 malware samples obtained from the VirusShare database [10]. Manavi and Hamzeh [11] presented a method for detecting ransomware using the PE header. They used a Convolutional neural network (CNN) to identify ransomware by converting the header bytes into 32*32 pixel images. The use of a header is advantageous, but transforming it into an image would necessitate the use of a network with additional layers in order to extract its features.

To detect ransomware, the authors [12] used a static method. They proposed a method that is based on the bytes extracted from the header of the executable file using LSTM network to build the detection model.

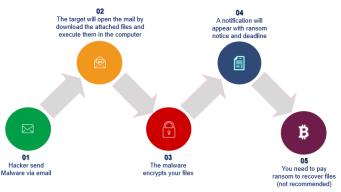


Fig. 1. Ransomware attack phases.

^{*}Corresponding Author.

The modification of the file header changes its structure. Therefore, they did the extraction of the executable file headers, then they processed the byte sequence that builds the file header with LSTM network, and they separated the ransomware samples, from the benign samples to form the template. With this technique, they managed to detect ransomware with 93.25% accuracy without running the program.

Subedi et al. [13] employed data mining techniques to recognize and detect ransomware families using both static and dynamic analysis at three different levels: assembly, function calls and library. They also created an analytical tool that uses reverse engineering to create signatures for identifying ransomware families. Arabo et al. [14] proposed a dynamic analysis approach to gather ransomware API properties, which is then utilized to test 9 Machine Learning classifiers and a neural network. The goal of this research is to understand the link between a process's behavior and its nature, to detect if it is a ransomware or not. With a detection rate of 75.01%, Random Forest surpasses other classifiers. The benefit of this technique is that it does not require a signature database, but rather a collection of ransomware and non-ransomware data. The detection rate of the classifiers may be better by improving the dataset.

Before encrypted files were moved to a backup disk, Lee et al. [15] utilized machine learning techniques to detect and classify infected files. The training step was implemented at the backup system according to their recommendation. It identified files from various users and file types, as well as determining file entropy thresholds. These thresholds were transmitted to client hosts in order to decide whether a new version of the file was encrypted or not. The authors in [16] suggest a two-stage mixed ransomware detection approach using Markov model with the Random Forest technique to detect ransomware. Random Forest has the best detection rate of 97.3%.

The paper [17] emphasizes the capabilities of behaviorbased detection mechanisms to identify crypto ransomware, demonstrating the limitations of signature-based detection approaches. In [18], Nieuwenhuizen proposed a ransomware detection scheme using behavior analysis and machine learning. Although the specific features were not revealed, their created feature set included properties such as payload persistence, anti-system restoration, stealth methods, environment mapping, network traffic, and privilege elevation that were extracted from the behavior of a malicious set up. Author employed the support vector machine (SVM) method as the classification technique in addition to the behavioral features related with data transformation behavior, such as huge file encryption.

The effect of certain ransomware families on the Windows platform is demonstrated and analyzed by Mohammad [19]. He deduces that most families of ransomware behave in a similar way when it comes to affect file system and registry entities. Furthermore, all types of ransomware generate files in the Windows system files and rename other files. To do the experiments, the author used Windows 7, Oracle VirtualBox VM, Cuckoo sandbox, and Virtual windows 10. The author concludes that monitoring system file and registry activities can protect against ransomware. He also mentions that Windows 10 is more effective than Windows 7 regarding malware. The best method to follow as a recommendation is to regularly back up company or individual data.

III. METHODOLOGY

As mentioned at the beginning, our goal is to detect whether a file is suspicious or not (regardless of its content), from its header which will be identified from its extension. This leads us to detect ransomware from encrypted files in real time.

It is well known that each extension has a fixed header according to the standards. If the header differs from the standard state, we deduce that it is suspicious.

To achieve our goal, we took several files with different extensions, if we take an extension, for example ".docx", and we open some files with the same extension using the Hexadecimal editor (Hex Editor Neo [20]), we found that they have the same signature, also called "Magic number".

Namely, each file extension (.docx, .pptx, .xls, .exe, .dll, .jpg, etc.) has a fixed and specific "Magic number".

Table I shows some files with different extensions and their signatures or Magic number, for clear and normal files.

According to a deep study on Microsoft Office files, we notice that their signature is different, the "x" added at the end made many differences. If we take the extensions .doc and .docx (the same thing for .xls, .ppt/.xlsx, .pptx), the differences are seen in Table II.

TABLE I. SIGNATURE OF FILES WITH DIFFERENT EXTENSIONS	
---	--

File extension	Hex signature	Size	ASCII Signature	
.doc,.ppt,.xls	D0 CF 11 E0 A1 B1 1A E1	8 Bytes	ÐÏ.à;±.á	
.docx , .pptx , .xlsx	50 4B 03 04 14 00 06 00	8 Bytes	РК	
.pdf	25 50 44 46	4 Bytes	%PDF-	
.png	89 50 4E 47 0D 0A 1A 0A	8 Bytes	.PNG	
.jpg	FF D8 FF E0	4 Bytes	ÿØÿà	
.dll	4D 5A 90 00	4 Bytes	MZ	
.exe	4D 5A	2 Bytes	– MZ	

 TABLE II.
 DIFFERENCE BETWEEN .DOC AND .DOCX

	DOC	DOCX
Version	Came in when the Microsoft Word was 1 st delivered and was utilized until 2003 variant of "Word".	Came in with Word 2007 and has been the default extension from then for all new Word versions.
Storage	A DOC is saved in a binary file that contains all the related formatting and relevant informations.	A DOCX file is actually a zip file with all XML files associated with the document.
File size	The DOC format has a <i>greater</i> size than the DOCX format.	The DOCX format has a <i>smaller</i> size than the DOC format.

To analyze files, we have taken some files (.docx, .doc, .pdf, .exe, .png) as an example, and see their structures (this study of files is very difficult to make because there is a lack of information on the structure of the files, for example, for the header "how many bytes occupied", for the contents "where does it begin?", etc.):

	Image: Standburger 0 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15																	
File Signature:	0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
- Local file header	0	50	4b	03	04	14	00	06	00	108	00	00	00	21	00	2da	30	PK
signature: => 50 48 03 03	16	a5	23	96	01	00	00	4f3	05	00	00	13	00	808	02	5b	43	¥#ó[C
- Version:	32	6f	6e	74	65	6e	74	5f	54	79	70	65	73	5d	2e	78	6d	ontent_Types].xm
=> 14 00 = 20 (2.0)	48	6c	20	a2	04	02	28	a0	00	02	00	00	00	00	00	00	00	1 ¢(
 General purpose bit flags: 	64	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
=> 06 00	80	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
	96	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
	112	00	nn	00	00	00	nn	00	nn	00	nn	00	nn	00	nn	00	00	
(1) Me	thod Co	mpre	ssio	on				(2)	CR	C (C	/clio	Re	dun	dan	cy (Chec	k) 4	Bytes
(4) Und	ompres	sed S	Size					(3)	Со	mpre	sse	d Si	ze					
	 (4) Uncompressed Size (3) Compressed Size (5) File Name length (13 00) Hex = (19) Dec =>> The file name "[Content types], xml" contains 19 characters 																	
(6) Extra Field Length (7) File name: [Content_Types].xml																		

Fig. 2. ".Docx" File signature.

From the article [21], the header is always at the beginning of the file and is exactly 512 bytes in length. Fig. 2 and Fig. 3 show you some information about the header of the ".docx" and ".doc" file, respectively.

		0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		The number of
Fix	ed File	0	d0	cf	11	e0	al	bl	la	el	1 00	00	00	00	00	00	00	00	0Ï.å;±.ć	sectors used by the sector allocation
Sig	nature	16	00	00	00	00	00	00	00	00	3e	00	03	00	fe	ff	09	00	þÿ	table is represented
		32	06	00	00	00	00	00	00	00	00	00	00	00	01	00	00	00		by 4 bytes. In this
		48	27	00	00	00	00	00	00	00	500	10	00	00	29	00	00	00	')	case, the SAT only
		64	01	00	00	00	fe	ff	ff	ff	00	00	00	00	26	00	00	00	þÿÿÿ	uses one sector.
		80	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	999999999999999999999	
		96	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	******	
		112	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	<i>3333333333333333333333</i>	
		128	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	<u> 9999999999999999999</u>	
		••••																		
		480	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	<u>9999999999999999999</u>	
		496	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	11	ff	ff	ff	ff	<i>8888888888888888888888</i>	
		512	ec	a5	cl	00	6b	e0	09	04	00	00	fO	12	bf	00	00	00	ì¥Á.kàð.;	
		528	00	00	00	30	00	00	00	00	00	80	00	00	01	80	00	00	0	
		544	0e	00	62	бa	62	бa	9a	ad	9a	ad	00	00	00	00	00	00	bjbjš-š	
		560	00	00	00	00	00	00	00	00	00	00	00	00	0c	04	16	00		
1) A u	inique id	entifie	r is s	tore	d in	16 by	/tes,	foll	owe	d by	4 byte	s (3	e 00	03 C	0) th	at ce	onta	in a r	evision number and a v	rersion number
2) The	e bye ord	er idei	ntifie	r is s	store	d in	2 by	tes.'	The l	oyte	seque	nce	"FE F	FF" s	nould	d alv	vays	be u	sed.	
3) 09	00 : Size	of Sec	tors (2^9	= 51	2)						10	5)4	bytes	con	tain	ing t	he mi	nimum size of standard	d streams.
	00 : Size												5) Th	e Se	ld of	the	1st	secto	r of the short-sector all	ocation table is
	e SecId of										bytes.	1							d by the number of sect	
The	e director	ry star	ts at	sect	or 3	9 (27	Hex	= 39	Dec	:)			SS	AT in	4 by	tes.	Here	the !	SAT starts at sector 41	and uses one sector

Fig. 3. ".doc" File signature (empty file).

Fig. 4 and 5 show you the structure and header of the ".pdf" and ".exe" files, respectively.

🔊 Survey.po	lf X																
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	25	50	44	46	2d	31	2e	34	0a	33	20	30	20	6f	62	6a	PDE-1.4.3 0 obj
16	0a	3c	3c	2f	54	79	70	65	20	2f	50	61	67	65	0a	2f	.<
32	50	61	72	65	6e	74	20	31	20	30	20	52	0a	2f	4d	65	Parent 1 0 R./Me
48	64	69	61	42	6f	78	20	5b	30	20	30	20	36	31	32	2e	diaBox [0 0 612.
64	30	30	20	37	39	32	2e	30	30	5d	0a	2f	52	65	73	6f	00 792.00]./Reso
80	75	72	63	65	73	20	32	20	30	20	52	0a	2f	41	6e	6e	urces 2 0 R./Ann
96	6f	74	73	20	5b	3c	3c	2f	54	79	70	65	20	2f	41	6e	ots [<
112	6e	6f	74	20	2f	53	75	62	74	79	70	65	20	2f	4c	69	not /Subtype /Li
128	6e	6b	20	2f	52	65	63	74	20	5b	32	36	35	2e	31	37	nk /Rect [265.17
144	20	36	31	31	2e	30	30	20	33	36	39	2e	30	30	20	35	611.00 369.00 5
160	39	36	2e	35	35	5d	20	2f	42	6f	72	64	65	72	20	5b	96.55] /Border [
176	30	20	30	20	30	5d	20	2f	41	20	3c	3c	2f	53	20	2f	0 0 0] /A <
192	55	52	49	20	2f	55	52	49	20	28	6d	61	69	6c	74	6f	URI /URI (mailto

Fig. 4. ".PDF" File Signature.

2.exe	×																	
0		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0		4d	5a	90	00	03	00	00	00	04	00	00	00	ff	ff	00	00	MZÿÿ
16		b8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	
32		00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
48		00	00	00	00	00	00	00	00	00	00	00	00	18	01	00	00	
64		0e	lf	ba	0e	00	b4	09	cd	21	b8	01	4c	cd	21	54	68	°´.Í!,.LÍ!Th
80		69	73	20	70	72	6f	67	72	61	6d	20	63	61	6e	6e	6f	is program canno
96		74	20	62	65	20	72	75	6e	20	69	6e	20	44	4f	53	20	t be run in DOS
112		6d	6f	64	65	2e	0d	0d	0a	24	00	00	00	00	00	00	00	mode\$

Fig. 5. ".exe" file signature.

IV. RESULTS AND DISCUSSION

We took a corpus[22] that contains a large number of files with different extensions, and I encrypted them with a python program, adding to its files an '.enc' extension to make the difference between a clear file and an encrypted file. As an example, I took four files for each different extension (.doc, .docx, .ppt, .pdf); we got the following result:

l.doc x																			
🛾 🖬 1.d	loc ×								_	🌁 🖿 🍮 1.c	loc.er	IC	· · L						
00000	0 1	2	3	4	5	6	7		~	00000000	0	1	2	3	4	5	6	7	
00000	d0 cf	11	e0	al	bl	la (e1	Ï.à;±.á		00000000	2f	2e	02	89	56	38	dd	aa	%V8Ý*
80000	00 00	00	00	00	00	00	00			00000008	01	e4	71	2c	a4	3f	31	0a	.äq,¤?1.
00010	00 00	00	00	00	00	00	00			00000010	5f	3d	93	c3	de	ee	82	00	_="ÃÞî,.
00018	3e 00	03	00	fe	ff	09	00	>þÿ		00000018	83	lb	bb	74	cl	2a	ee	a8	f.»tÁ*î″
00020	06 00	00	00	00	00	00	00			00000020	77	f3	37	be	aa	0d	7b	b4	wó7¾ª.{′
00028	00 00	00	00	01	00	00	00			00000028	c4	le	ef	2c	76	72	d3	0d	Ä.ï,vrÓ.
00030	4b 00	00	00	00	00	00	00	К		00000030	cb	71	aa	53	2a	ae	de	e7	Ëq²S*®₽¢
2.0	doc 🗙									▲▶ 🗞2.0	loc.er	ic	×						
00000	0 1	2	3	4	5	6	7		~	00000000	0	1	2	3	4	5	6	7	
00000	d0 cf	11	e0	al	bl	1a	e1	Ĭ.å;±.á		00000000	a7	c5	de	d5	24	9d	fa	e2	ÅÞĈ\$ úâ
80000	00 00	00	00	00	00	00	00			80000000	8c	23	a9	68	66	e7	4f	65	€ #©hfç0e
00010	00 00	00	00	00	00	00	00			00000010	ba	88	d4	3a	61	bc	d5	3d	°^ô:a%Õ=
00018	3e 00	03	00	fe	ff	09	00	>þÿ		00000018	27	28	09	41	32	23	51	dО	'(.A2#Q.
00020	06 00	00	00	00	00	00	00			00000020	16	f6	fd	04	bd	1b	15	21	.öý.¥!
00028	00 00	00	00	01	00	00	00			00000028	lb	68	7f	10	7f	d8	8d	d8	.h].]ØØ
00030	6c 00	00	00	00	00	00	00	1		00000030	8a	7e	de	bd	0c	d 8	eb	13	Š~₽¾.Øë.
🖬 3.d	loc 🗙									▲► Sidoc.enc X									
00041	0 1	2	3	4	5	6	7	-	~	00000062	0	1	2	3	4	5	6	7	
00000	d0 cf	11	e0	al	bl	la e	-1	ÐÏ.ð;±.á		00000000	1e	0e	ac	97	01	ad	d7	0b	×.
80000	00 00	00	00	00	00	00 0	00			00000008	6f	4c	fO	fd	8e	d7	93	79	olðýŽ×"y
00010	00 00	00	00	00	00	00 0	00			00000010	£7	сс	98	99	d0	2c	18	73	÷Ì~≞⊕,.s
00018	3e 00	03	00	fe	ff	09 (00	>þÿ		00000018	72	al	lc	d6	6d	ce	cb	09	r;.ÖmÎË.
00020	06 00	00	00	00	00	00 0	00			00000020	5f	fa	76	b3	9a	db	dc	5d	_úv³šÛÜ]
00028	00 00	00	00	01	00	00 0	00			00000028	2c	a6	41	65	5b	82	c4	2e	, Ae[,Ä.
00030	68 00	00	00	00	00	00 0	00	h		00000030	9a	9f	80	0a	le	de	96	fb	šŸ€Þ-û

Fig. 6. The difference between encrypted and clear file headers [.doc].

For the files "*.doc " (in Fig. 6), those on the left are clear files, their header should be normal [D0 CF 11 E0 A1 B1 1A E1]. While on the right, you see that there is an extension added at the end "*.doc.enc ", this means that they are encrypted files (the encrypted file of each clear file, e.g. "1.doc.enc" is the encrypted file of "1.doc"), and even their header is different. What is relevant is that each encrypted file has a different header from the other file, we have [2F 2E 02 89 56 38 DD AA], [A7 C5 DE D5 24 9D FA E2], etc.

The same thing for the files "*.docx " (in Fig. 7), those on the left are clear files, their header should is [50 4B 03 04 14 00 06 00], while on the right you see the encrypted files and even their header is different. We have [A1 D1 05 C2 6C 8B 1D 19], [EF 34 21 9F FE 78 65 FC], etc.

💼 Exe	ercice.doc	××					Ex	ercice.docx.enc 🗙
0000	0 1	2 3	4	5	67	^	00000000	0 1 2 3 4 5 6 7
0000	50 4b	03 0	4 14	00	06 00	K	00000000	dl 05 c2 6c 8b ld 19 N.Âl<
8000	08 00	00 0	0 21	00	61 44	!.aD	80000000	5a 98 4c cc 0f 58 8b 46 Z~LÌ.X <f< td=""></f<>
0010	05 d5	a5 0	1 00	00	1b 07	.õ¥	00000010	d8 d4 9b ab fb d5 14 bf 00>«ûÕ.;
0018	00 00	13 0	08 0	02	5b 43	[C	00000018	78 df a7 b1 20 f1 a7 33 xB§± ñ§3
0020	6f 6e	74 6	5 6e	74	5f 54	ontent_T	00000020	c4 18 c8 b0 d6 59 37 d5 Ä.ȰÖY7Õ
0028	79 70	65 7		2e	78 6d	ypes].xm	00000028	02 da 40 34 bl 0f e8 el .Ú@4±.èá
0030	6c 20	a2 0	4 02	28	a0 00	l ¢(.	00000030	ee 13 d4 ad 39 41 72 cd î.Ô-9ArÍ
RN 🕼	A.docx	×					RN	IA.docx.enc X
0000	0 1	2 3	4	5	67	<u>^</u>	00000000	0 1 2 3 4 5 6 7
0000	50 4b	03 0	4 14	00	06 00	к	00000000	34 21 9f fe 78 65 fc 4!Ÿþxeü
8000	08 00	00 0	0 21	00	9e 1c	!.ž.	00000008	fd b6 e9 eb 88 37 e8 bd ý¶éë^7è%
0010	dc 9a	8e 0	1 00	00	c2 05	ÜšŽÂ.	00000010	a4 e7 a4 a6 97 8a 28 da ¤ç¤¦—Š(Ú
0018	00 00	13 0	08 0	02	5b 43	[C	00000018	71 4c db e5 ad 4d 36 d4 qLÛå-M6Ô
0020	6f 6e	74 6	5 6e	74	5f 54	ontent_T	00000020	d2 a8 ae ed bd ec d2 f5 Ò"@i%iÒõ
0028	79 70	65 7		2e	78 6d	ypes].xm	00000028	d5 9e e9 fa ec 6e eb al Õžéúìnë;
0030	6c 20	a2 0	4 02	28	a0 00	1 ¢(.	00000030	9a 5c fc 38 06 e6 48 07 š\ü8.æH.
💼 séc	curité.doc	××					🔺 🕨 🗞 sé	curité.docx.enc 🗙
0000	0 1	2 3	4	5	67		00000000	0 1 2 3 4 5 6 7
0000	50 4b	03 0	4 14	00	06 00	к	00000000	7f c5 6d a2 89 cd f9
8000	08 00	00 0	0 21	00	09 24	\$	80000008	46 43 ef b4 3c a7 30 0a FCï′<§0.
0010	87 82	81 0	1 00	00	8e 05	ŧ,Ž.	00000010	2c e8 3a ae c7 a0 48 13 ,è:@Ç H.
0018	00 00	13 0	08 0	02	5b 43	[C	00000018	a8 b5 0f ab 0f 34 65 a4 "µ.«.4e¤
0020	6f 6e	74 6	5 6e	74	5f 54	ontent_T	00000020	27 bl 88 35 54 b2 a5 15 '±^5T'¥.
0028	79 70	65 7		2e	78 6d	ypes].xm	00000028	66 dd ad e6 79 a6 62 d2 fÝ-æy¦bÒ
0030	6c 20	a2 0	4 02	28	a0 00	1 ¢(.	00000030	dd ee 77 d5 d9 61 2d e5 ÝîwÕÙa-å
Vic 🖉	de.docx	×					🔺 🕨 📚 Vie	de.docx.enc X
Lana I							00000000	
0000	0 1	2 3	4	5	67	^	00000000	0 1 2 3 4 5 6 7
0000	0 1	2 3		5 00	6 7 06 00	K	00000000	0 1 2 3 4 5 6 7 3e 2b 9e 85 74 78 d7
			4 14	-	_	K		
0000	50 4b	03 0	4 14 0 21	00	06 00		00000000	3e 2b 9e 85 74 78 d7
0000	08 00	03 0	4 14 0 21 1 00	00	06 00 df a4	!.B×	00000000	3e 2b 9e 85 74 78 d7 7d 54 67 10 ba b0 a6 2f]Tg.°°// 6d 3a fe 1a 12 33 2b 73 mip3+s ac 19 90 46 99 aa bc ao F ^{mana}
0000 0008 0010 0018 0020	4b 08 00 d2 6c 00 00 6f 6e	03 0 00 0 5a 0 13 0 74 6	4 14 0 21 1 00 0 08 5 6e	00 00 00 02 74	06 00 df a4 20 05 5b 43 5f 54	!.B¤ ÒlZ[C ontent_T	00000000 00000008 00000010 00000018 00000020	3e 2b 9e 85 74 78 d7)+± txx 7d 54 67 10 ba b0 a6 2f]Tg.°°¦/ 6d 3a fe 1a 12 32 b7 mip3*s ac 19 90 64 98 ab a0 F***** 1b 69 bb c9 df a0 ******
0000 0008 0010 0018	08 00 08 00 d2 6c 00 00	03 0 00 0 5a 0 13 0	4 14 0 21 1 00 0 08 5 6e 3 5d	00 00 00 02	06 00 df a4 20 05 5b 43	!.B¤ òlz[C	00000000 00000008 00000010 00000018	3e 2b 9e 85 74 78 d7 7d 54 67 10 ba b0 a6 2f]Tg.°°// 6d 3a fe 1a 12 33 2b 73 mip3+s ac 19 90 46 99 aa bc ao F ^{mana}

Fig. 7. The difference between encrypted and clear file headers [.DOCX].

🛃 1.p	odf X			▲ ► (🇞 1.p	odf.enc 🗙
0000	0 1 2 3	4 5 6 7	•	00000000	0 1 2 3 4 5 6 7
0000	25 50 44 46	2d 31 2e 34	PDE-1.4	00000000	89 7b b4 c2 cb 97 35 ts { ÂĔ-5
8000	0d 25 e2 e3	cf d3 0d 0a	.%âãïó	00000008	11 29 95 b2 ba ff 40 95 .).*°ÿ@.
0010	31 39 20 30	20 6f 62 6a	19 0 obj	00000010	e3 6f 0d 1f 29 ae f5 d7 ão)®ő×
0018	0d 3c 3c 20	0d 2f 4c 69	.<< ./Li	00000018	23 3a 39 78 fc 60 68 cf #:9xü`hl
0020	6e 65 61 72	69 7a 65 64	nearized	00000020	fa 6c 23 f0 3d 5f la e2 úl#ð=â
0028	20 31 20 0d	2f 4f 20 32	1 ./0 2	00000028	11 8f d3 00 06 58 1d b5 . ÓX.p
0030	32 20 0d 2f	48 20 5b 20	2 ./H [00000030	95 lc cf 51 f7 f4 21 ab •.ĨQ÷ô!«
🔎 2.p	odf 🗙			▲ ► <u></u> <u>8</u> 2.p	odf.enc 🗙
0000	0 1 2 3	4 5 6 7	•	00000000	0 1 2 3 4 5 6 7
0000	25 50 44 46	2d 31 2e 33	PDE-1.3	00000000	3 70 7a 0a 68 28 36 e7 pz.h(60
8000	0d 25 e2 e3	cf d3 Od Oa	.§âãïó	80000000	2e 59 fd e3 1f bc f0 52 .Yýã.w6F
0010	32 31 20 30	20 6f 62 6a	21 0 obj	00000010	91 67 61 80 ef d5 a2 35 `gaە͢5
0018	0d 3c 3c 20	0d 2f 4c 69	.<< ./Li	00000018	8c 98 be 76 eb 4d e7 3f CE**vëMç?
0020	6e 65 61 72	69 7a 65 64	nearized	00000020	bl 78 e5 8c dl 01 2e f8 ±xå@Ñø
0028	20 31 20 0d	2f 4f 20 32	1 ./0 2	00000028	6b 39 47 60 72 04 82 6a k9G`r.,j
0030	33 20 0d 2f	48 20 5b 20	3 ./H [00000030	ba 69 ef 86 84 16 70 c3 °iït".pÅ
🔊 3.p	odf 🗙			● ► (& 3.p	odf.enc X
					anche A
0000	0 1 2 3	4 5 6 7	^	00000000	0 1 2 3 4 5 6 7
		4 5 6 7 2d 31 2e 34	PDE-1.4		
0000	0 1 2 3		PDE-1.4 .\$âãIÓ	00000000	0 1 2 3 4 5 6 7
0000	0 1 2 3	2d 31 2e 34		00000000	0 1 2 3 4 5 6 7
0000	0 1 2 3 50 44 46 0d 25 e2 e3	2d 31 2e 34 cf d3 0d 0a	.%âãÏÓ	000000000000000000000000000000000000000	0 1 2 3 4 5 6 7 0 0 e9 dc 5b 41 9a 4btt (Aär 29 2c 11 db c3 93 ca d6),.0ä*bt df c2 30 b9 16 b7 f8 6a 8å0eg 07 dd f8 73 b9 aa 46 5att (S*s*tz)
0000 0000 0008 0010	0 1 2 3 50 44 46 0d 25 e2 e3 36 20 30 20	2d 31 2e 34 cf d3 0d 0a 6f 62 6a 0d	.%âãÏÓ 6 0 obj.	000000000000000000000000000000000000000	0 1 2 3 4 5 6 7 0 0 e9 dc 5b 41 9a 4b
0000 0000 0008 0010 0018	0 1 2 3 50 44 46 0d 25 e2 e3 36 20 30 20 3c 3c 2f 4c	2d 31 2e 34 cf d3 0d 0a 6f 62 6a 0d 69 6e 65 61	.%âãÏÓ 6 0 obj. < <td>00000000 00000000 00000008 00000010 00000018</td> <td>0 1 2 3 4 5 6 7 0 0 e9 dc 5b 41 9a 4btt (Aär 29 2c 11 db c3 93 ca d6tt (Aär df c2 30 b9 16 b7 f8 6a 8Å0¹tt) 07 dd f8 73 b9 aa 46 5att (Strester)</td>	00000000 00000000 00000008 00000010 00000018	0 1 2 3 4 5 6 7 0 0 e9 dc 5b 41 9a 4btt (Aär 29 2c 11 db c3 93 ca d6tt (Aär df c2 30 b9 16 b7 f8 6a 8Å0 ¹ tt) 07 dd f8 73 b9 aa 46 5att (Strester)
0000 0000 0008 0010 0018 0020	0 1 2 3 50 44 46 0d 25 e2 e3 36 20 30 20 3c 3c 2f 4c 72 69 7a 65	2d 31 2e 34 cf d3 0d 0a 6f 62 6a 0d 69 6e 65 61 64 20 31 2f	.§âãÏÓ 6 0 obj. <rized 1/	00000000 00000000 00000008 00000010 00000018 00000018	0 1 2 3 4 5 6 7 0 0 e9 dc 5b 41 9a 4b
0000 0008 0010 0018 0020 0028	0 1 2 3 50 44 46 0d 25 e2 e3 36 20 30 20 3c 3c 2f 4c 72 69 7a 65 4c 20 32 31 4f 20 38 2f	2d 31 2e 34 cf d3 0d 0a 6f 62 6a 0d 69 6e 65 61 64 20 31 2f 36 30 34 2f	.%ääÏÓ 6 0 obj. <rized 1/ L 21604/	00000000 00000000 00000008 00000010 00000018 00000020 00000028	0 1 2 3 4 5 6 7 0 0 de 9 dc 15 41 9a 4b .et/(A8) 29 2c 11 db c3 93 ca d6) .0% 76 df c2 30 b3 16 b7 f8 6a bÅ ^{0.1} eg 07 dd f8 73 b9 aa 46 5a ·¥s ^{0.5} F2 e6 a5 ad da 970 46 ae aV-ODPAE 33 34 da 69 59 c6 32CD.v12 f2 e0 bb 80 as 9d dd àAP=E Y f3
0000 0008 0010 0018 0020 0028	0 1 2 3 50 44 46 0d 25 e2 e3 36 20 30 20 3c 3c 2f 4c 72 69 7a 65 4c 20 32 31 4f 20 38 2f	2d 31 2e 34 cf d3 0d 0a 6f 62 6a 0d 69 6e 65 61 64 20 31 2f 36 30 34 2f	.%ääÏÓ 6 0 obj. <rized 1/ L 21604/	00000000 00000000 00000008 00000010 00000018 00000020 00000028 00000028	0 1 2 3 4 5 6 7 0 0 de 9 dc 15 41 9a 4b .et/(A8) 29 2c 11 db c3 93 ca d6) .0% 76 df c2 30 b3 16 b7 f8 6a bÅ ^{0.1} eg 07 dd f8 73 b9 aa 46 5a ·¥s ^{0.5} F2 e6 a5 ad da 970 46 ae aV-ODPAE 33 34 da 69 59 c6 32CD.v12 f2 e0 bb 80 as 9d dd àAP=E Y f3
0000 0008 0010 0018 0020 0028 0030	0 1 2 3 50 44 46 0d 25 e2 e3 36 20 30 20 3c 3c 2f 4c 72 69 7a 65 4c 20 32 31 4f 20 38 2f off X	2d 31 2e 34 cf d3 0d 0a 6f 62 6a 0d 69 6e 65 61 64 20 31 2f 36 30 34 2f 45 20 31 37	.%ääÏÓ 6 0 obj. <rized 1/ L 21604/	00000000 00000000 00000000 00000010 000000	0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 0 e9 dc 50 41 9a 4b€[AåF 29 2c 11 db c3 93 ca d6),.0åF45 df c2 30 09 16 b7 16 6a ÅÅ0 ¹ .og 07 dd f8 73 b9 aa 46 5a∱res ¹ FZ e6 a5 ad da d9 70 46 ae e¥-0ĎDF 33 33 43 dc 9b 0a 59 c6 33CDF f2 e0 de bb 80 aa 9d dd òàĐ≈€* 5 odf.ecc X
0000 0008 0010 0018 0020 0028 0030	0 1 2 3 50 44 46 0d 25 e2 e3 36 20 30 20 3c 3c 2f 4c 72 69 7a 65 4c 20 32 31 4f 20 38 2f o 1 2 3	2d 31 2e 34 cf d3 0d 0a 6f 62 6a 0d 69 6e 65 61 64 20 31 2f 36 30 34 2f 45 20 31 37	.%ääIÓ 6 0 obj. <rized 1/ L 21604/ O 8/E 17	00000000 00000008 00000010 00000018 00000020 00000028 00000028 00000030	0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 29 2c 11 db c3 93 ca d6), 0Å ±Č(ÅÅ 4 c2 3 0b 9 16 b7 16 6 8 ÅÅ ^{0.} • og 07 dd f8 73 b9 aa 46 5a × × × × × × × × × × × × × × × × ×
0000 0008 0010 0018 0020 0028 0030 0028 0030	0 1 2 3 50 44 46 0d 25 e2 e3 36 20 30 20 3c 3c 2f 4c 72 69 7a 65 4c 20 32 31 4f 20 38 21 4f 20 38 21 0 1 2 3 50 44 46	2d 31 2e 34 cf d3 0d 0a 6f 62 6a 0d 69 6e 65 61 64 20 31 2f 36 30 34 2f 45 20 31 37 4 5 6 7 2d 31 2e 34	.%aãIÓ 6 0 obj. <rized 1/ L 21604/ O 8/E 17	00000000 00000008 00000010 00000018 00000020 00000020 00000020 ■► &4.p 00000000 00000000	$\begin{array}{c ccccccccccccccccccccccccccccccccccc$
0000 0008 0010 0018 0020 0028 0030 (A.F 0000 0000 0000	0 1 2 3 50 44 46 04 25 e2 e3 36 20 30 20 35 3c 2f 4c 72 69 7a 65 4c 20 32 21 4f 20 38 2f 0 1 2 3 0 44 46 0 42 5 e2 e3 0 50 44 46 0 4 25 e2 e3 0 50 44 46 0 4 2 5 e2 e3 0 50 50 44 46 0 50 50 50 50 50 0 50 50 50 50 0 50 50 50 50 0	2d 31 2e 34 cf d3 0d 0a 6f 62 6a 0d 69 6e 65 61 64 20 31 2f 45 20 31 37 4 5 6 7 2d 31 2e 34 cf d3 0d 0a	.%ääïó 6 0 obj. <rized 1/ L 21604/ O 8/E 17 PDE-1.4 .%ääïó	00000000 0000008 0000010 0000018 00000018 00000028 00000028 00000030 4 b & 4.p 00000000 00000000	0 1 2 3 4 5 6 7 10 0d e9 dc 55 41 9a 4b €(1ÅÅ£ 29 2c 11 db c3 93 ca d6 1, 0Å*Å 20 2c 11 db c3 93 ca d6 aÅÅ* 0 1 df 87 b9 aa 46 5a ½s***72 0 1 df 87 b9 aa 46 sa ½s****72 2 0 de b8 0 59 c6 32 ½s****72 2 0 de b8 80 a9 3d åb************************************
0000 0008 0010 0018 0020 0028 0030 (A 4,p 0000 0000 0000 0008 0010	0 1 2 3 0 25 e2 e3 3 20 20 44 46 0 25 e2 e3 3 2 0 20 3 2 2f 4c 7 2 69 7 a 65 4 20 3 2 31 4 f 20 38 2f 0 1 2 3 0 1 2 3 0 25 e2 e3 3 2 31 2 0 30 0 25 e2 e3 3 2 31 2 0 30 0 3 2 31 2 0 30 0 4 4 66 3 2 3 2 31 2 0 30 0 4 5 e2 e3 3 2 31 2 0 30 0 5 4 4 66 0 4 5 e2 e3 0 4 4 66 0 4 5 e2 e3 0 5 e2 e3 0 5 e2 e3 0 4 4 66 0 5 e2 e3 0 2 5 e2 e3 0 2 3 2 31 0 3 0 3 0 0 5 e2 e3 0 3 2 31 0 3 0 30 0 5 e2 e3 0 3 2 31 0 3 0 30 0 5 e2 e3 0 3 2 31 0 3 0 30 0 5 e2 e3 0 3 2 31 0 3 0 30 0 5 e3 0 5 e2 e3 0 5 e3	2d 31 2e 34 cf d3 0d 0a 6f 62 6a 0d 69 6e 65 61 64 20 31 2f 36 30 34 2f 45 20 31 37 4 5 6 7 2d 31 2e 34 2d 0d 0a 20 6f 62 6a	. %aāīć 6 0 obj. <rized 1/ L 21604/ O 8/E 17 PDE-1.4 .%aāīć 21 0 obj	00000000 00000008 00000019 00000018 00000028 00000028 000000030 ◀► ॆ ॆ ♣ 4.p 00000000 00000000 00000000	$\begin{array}{c ccccccccccccccccccccccccccccccccccc$
0000 0008 0010 0018 0020 0028 0030 0030 0030 0000 0000 000	0 1 2 3 0 42 62 62 63 36 20 30 20 36 20 32 21 47 20 38 2f 0 1 2 3 50 44 46 00 25 62 63 32 31 20 30 0 3 6 36 2f 1 2 3 1 2 5 1 2 3 1 2 5 1 2 3 1 2 5 1 2 5	2d 31 2e 34 cf d3 0d 0a 6f 62 6a 0d 69 6e 65 61 64 20 31 2f 36 30 34 2f 45 20 31 37 2d 31 2e 34 cf d3 0d 0a 20 6f 62 6a 40 69 6e 65	. \$&&IO. 6 0 obj. <rized 1/ L 21604/ 0 8/E 17 PDB-1.4 . \$&&IO. 21 0 obj. .< <td>00000000 00000000 00000000 00000018 00000020 00000020 00000020 00000000 000000</td> <td>$\begin{array}{c c c c c c c c c c c c c c c c c c c$</td>	00000000 00000000 00000000 00000018 00000020 00000020 00000020 00000000 000000	$\begin{array}{c c c c c c c c c c c c c c c c c c c $
0000 0008 0010 0018 0020 0028 0030 0028 0030 0028 0030 0000 000	0 1 2 3 0 44 46 0 25 e2 e3 3 20 30 20 3 5 21 4c 2 0 7a 65 4 20 32 31 4 f 20 38 2f 0 1 2 3 0 1 2 3 0 25 644 46 0 25 e2 e3 3 2 1 20 30 0 3 c 3 c 2f 6 1 72 69 7a 6 20 32 31 4 f 20 38 2f 0 1 2 3 1 3	$\begin{array}{cccccccccccccccccccccccccccccccccccc$. \$8816 6 0 obj. <Tized 1/ L 21604/ 0 8/E 17 PDB-1.4 . \$8816 21 0 obj. .<arized 1	00000000 00000000 00000010 00000010 000000	0 1 2 3 4 5 6 7 1 0 d e9 dc Sh 41 9a 4b 29 2c 11 db c3 93 ca d6 p., 0ÅrÅt 29 2c 11 db c3 93 ca d6 p., 0ÅrÅt df c2 30 b9 16 b7 f8 fa bÅr ¹ .cg 0 1 f8 pa a4 65 a a ¹ V ⁰ Dp Ea 3 34 da d9 70 da ae a ¹ V ⁰ Dp Ea 2 0 de b8 0a 59 c6 32CD.v13 f2 ab#** 3 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 5

Fig. 8. The difference between encrypted and clear file headers [.PDF].

As you can see in Fig. 6, Fig. 7 and Fig. 8, the signature of a clear file and its encrypted is not the same; the case is the same for the other extensions. We also notice that the signature is fixed for any clear file (with different extension), but for encrypted files, it is not fixed and differs between each file.

The program is done by Python language to make this study and detect if a file is suspicious or normal from its signature. Each extension has a "Magic Byte". We instantiated our dataset by creating a dictionary with the file extension as a key and its "Magic byte" as value, and then we analyze the file. If the file does not contain the corresponding signature, i.e. it has a different header than the one presented in our dataset; we deduce that it is a suspect file. We have also dealt with the case of a file without extension, if we give it to our program, it analyzes the header and if it does not find the corresponding signature, it sends us back that it's a suspicious file, otherwise, if everything is normal the result is: "This is a benign file, its extension is: ... ". Fig. 9 shows the result of a file without extension.

28 27 28 29	<pre>27 with open('File', 'rb') as fd: 28 file_head = fd.read(max_read_size) 29</pre>																
<	٢																
_	Console 2/A 🗵																
This i) ign	Fil	e, I	it's	ex	ten:	ison	is:	doo	cx.					
· 🗍 File	e X)															
00000	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f	
00000	50	4b	03	04	14	00	06	00	08	00	00	00	21	00	df	a4	BK
00010	d2	6c	5a	01	00	00	20	05	00	00	13	00	08	02	5b	43	ò1z[C
00020	6f	6e	74	65	6e	74	5f	54	79	70	65	73	5d	2e	78	6d	ontent_Types].xm

Fig. 9. Analysis of a file without extension.

Fig. 10 shows you that we have performed an analysis for several files with different extensions.

<pre>27 with open('logo.jpg', 'rb') as fd: 28 file_head = fd.read(max_read_size) 29</pre>	<pre>with open('Manifestation.xlsx', 'rb') as fd: file_head = fd.read(max_read_size) 29</pre>
<	<
	Console 2/A 🗵
This is a <mark>benign</mark> File, It's extenison is: jpg	This is a <mark>benign</mark> File, It's extenison is: xlsx
27 27 28 29 5 5 5 5 5 5 5 5 5 5 5 5 5	<pre>20 27 with open('cours.pptx', 'rb') as fd: 28 29 29 29 20 20 20 20 20 20 20 20 20 20 20 20 20</pre>
	<
Console 2/A 🔀	Console 2/A 🔀
Fichier_Word') This is a <mark>benign</mark> File, It's extenison is: <mark>pdf</mark>	This is a <mark>benign</mark> File, It's extenison is: pptx
with open(' <u>file3.exe'</u> , 'rb') as fd: 28 file_head = fd.read(max_read_size)	<pre>2b 27 with open('<u>Evaluation.doc'</u>, 'rb') as fd: 28 file_head = fd.read(max_read_size) 29</pre>
<	< .
	Console 2/A 🗵
This is a benign File, It's extenison is: exe	This is a benign File, It's extenison is: doc

Fig. 10. Analysis of several normal files with different extensions.

If we take the example in Fig. 11, you can see that the result is "this is a suspicious file", even though the file has the extension ".doc". In effect, sometimes attackers send files that look normal with a legal extension, while the file is infected by the ransomware, so as you can see, our program perfectly analyzes the header of the given file identifying its signature, and it found that its signature does not match to the normal signatures.

<pre>27 with open('<u>FileS.doc'</u>, 'rb') as fd: 28 file_head = fd.read(max_read_size)</pre>														
<														
Console 1/A 🗵														
It's a suspicious file														
■▶ 🗑 FileS.doc 🗙														
00000000	00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f													
00000000	2f 2e 02 89 56 38 dd aa 01 e4 71 2c a4 3f 31 0a / XV8Ý*.äg,¤?1.													
00000010	5f 3d 93 c3 de ee 82 00 83 lb bb 74 cl 2a ee a8 ="ÃÞî,.f.»tÁ*î"													

Fig. 11. Analysis of a suspicious file.

We have tested our program on files encrypted by Ransomware with ".lmas", as you can see in Fig. 12, we have taken as an example the file "formation.xlsx.lmas", the ".lmas" extension is added after ".xlsx" extension. We got the followed result:

<pre>27 with open('formation.xlsx.lmas', 'rb') as fd: 28 file_head = fd.read(max_read_size) 29</pre>																	
<																	
	onsol	e 1/4															
It's a suspicious file																	
		<u> </u>															
formation.xlsx.lmas																	
0000	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f	
0000	50	4b	03	04	14	29	b0	98	9b	02	ed	e2	c3	be	cf	73	P <mark>K)°</mark> °>.íâþÏ
0010	32	d3	10	7a	af	08	ae	lf	8b	6d	46	46	9c	12	16	2d	2Ó.z [−] .@. <mffœ< td=""></mffœ<>
0020	£ A	00	bb	Ob	-	5 f	-6	16	e6	c/	fQ	4b	~9	78	43	60	ô~».Ä æFæÄùKÉxÓ

Fig. 12. Analysis of a file encrypted by Ransomware.

As you can see, Ransomware infects the file «formation.xlsx», it is encrypted and the attacker has added the extension ".lmas" to the file.

We know that "xlsx" has a fixed signature (see Table I); in the Fig. 12, we can see that the first 4 bytes are similar to the first 4 bytes of the normal xlsx file (50 4B 03 04), but the difference is in the next 4 bytes. Therefore, our program was able to detect that this file is encrypted by ransomware so it is a suspicious file without opening it.

V. CONCLUSION

In this work, we have made a Python program that allows to detect a suspicious file from another normal one, we started by studying the header of files of different extensions separately, later we extracted the header of each file and compared the headers of a normal file with another encrypted one. With this study, we could deduce that a normal file has a fixed and unchangeable extension, once it is changed the file is suspicious.

In the upcoming work, we will conduct a dynamic analysis by executing ransomware files in a simulated environment. This will allow us to extract ransomware encrypted files and analyze them in order to develop and implement our own neural network. This network will be trained to identify ransomware files by first learning the characteristics extracted from the ransomware encrypted files, and then using that knowledge to detect ransomware when a "vulnerable" file is downloaded onto a victim's device.

REFERENCES

- « Cyberattacks 2021: Statistics From the Last Year », Spanning, 18 janvier 2022. https://spanning.com/blog/cyberattacks-2021-phishingransomware-data-breach-statistics/ (consulté le 7 mars 2022).
- [2] S. Kok, A. Abdullah, N. Jhanjhi, et M. Supramaniam, « Ransomware, threat and detection techniques: A review », Int. J. Comput. Sci. Netw. Secur, vol. 19, no 2, p. 136, 2019.
- [3] C. V. Bijitha, R. Sukumaran, et H. V. Nath, « A Survey on Ransomware Detection Techniques », in Secure Knowledge Management In Artificial Intelligence Era, vol. 1186, S. K. Sahay, N. Goel, V. Patil, et M. Jadliwala, Éd. Singapore: Springer Singapore, 2020, p. 55-68. doi: 10.1007/978-981-15-3817-9_4.

- [4] C. Beaman, A. Barkworth, T. D. Akande, S. Hakak, et M. K. Khan, « Ransomware: Recent advances, analysis, challenges and future research directions », Computers & Security, vol. 111, p. 102490, déc. 2021, doi: 10.1016/j.cose.2021.102490.
- [5] S. I. Bae, G. B. Lee, et E. G. Im, « Ransomware detection using machine learning algorithms », Concurrency Computat Pract Exper, vol. 32, no 18, sept. 2020, doi: 10.1002/cpe.5422.
- [6] S. Poudyal, K. D. Gupta, et S. Sen, « PEFile Analysis: A Static Approach To Ransomware Analysis », The International Journal of Forensic Computer Science, vol. 14, p. 34-39, oct. 2019, doi: 10.5769/J201901004.
- [7] T. Rezaei et A. Hamze, « An Efficient Approach For Malware Detection Using PE Header Specifications », in 2020 6th International Conference on Web Research (ICWR), Tehran, Iran, avr. 2020, p. 234-239. doi: 10.1109/ICWR49608.2020.9122312.
- [8] V. Verma, S. K. Muttoo, et V. B. Singh, « Detection of Malign and Benign PE Files Using Texture Analysis », in Information Systems Security, Cham, 2020, p. 253-266. doi: 10.1007/978-3-030-65610-2_16.
- [9] I. Abdessadki et S. Lazaar, « A New Classification Based Model for Malicious PE Files Detection », International Journal of Computer Network and Information Security, vol. 11, p. 1-9, juin 2019, doi: 10.5815/ijcnis.2019.06.01.
- [10] « VirusShare.com ». https://virusshare.com/ (consulté le 25 février 2022).
- [11] F. Manavi et A. Hamzeh, « A New Method for Ransomware Detection Based on PE Header Using Convolutional Neural Networks », in 2020 17th International ISC Conference on Information Security and Cryptology (ISCISC), Tehran, Iran, sept. 2020, p. 82-87. doi: 10.1109/ISCISC51277.2020.9261903.
- [12] F. Manavi et A. Hamzeh, « Static Detection of Ransomware Using LSTM Network and PE Header », in 2021 26th International Computer Conference, Computer Society of Iran (CSICC), Tehran, Iran, mars 2021, p. 1-5. doi: 10.1109/CSICC52343.2021.9420580.
- [13] K. P. Subedi, D. R. Budhathoki, et D. Dasgupta, « Forensic Analysis of Ransomware Families Using Static and Dynamic Analysis », in 2018 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, mai 2018, p. 180-185. doi: 10.1109/SPW.2018.00033.
- [14] A. Arabo, R. Dijoux, T. Poulain, et G. Chevalier, « Detecting Ransomware Using Process Behavior Analysis », Procedia Computer Science, vol. 168, p. 289-296, 2020, doi: 10.1016/j.procs.2020.02.249.
- [15] K. Lee, S.-Y. Lee, et K. Yim, « Machine Learning Based File Entropy Analysis for Ransomware Detection in Backup Systems », IEEE Access, vol. 7, p. 110205-110215, 2019, doi: 10.1109/ACCESS.2019.2931136.
- [16] J. Hwang, J. Kim, S. Lee, et K. Kim, « Two-Stage Ransomware Detection Using Dynamic Analysis and Machine Learning Techniques », Wireless Pers Commun, vol. 112, no 4, p. 2597-2609, juin 2020, doi: 10.1007/s11277-020-07166-9.
- [17] P. S. Goyal, A. Kakkar, G. Vinod, et G. Joseph, « Crypto-ransomware detection using behavioural analysis », in Reliability, Safety and Hazard Assessment for Risk-Based Technologies, Springer, 2020, p. 239-251.
- [18] D. Nieuwenhuizen, « A behavioural-based approach to ransomware detection », Whitepaper. MWR Labs Whitepaper (2017), p. 20.
- [19] A. Mohammad, Analysis of Ransomware on Windows platform, International Journal of Computer Science and Network Security 20.6 (2020): 21-27. 2020. doi: 10.13140/RG.2.2.11150.59202.
- [20] « Free Hex Editor: Fastest Binary File Editing Software. Freeware. Windows ». https://www.hhdsoftware.com/free-hex-editor (consulté le 25 février 2022).
- [21] D. Rentz, « The Microsoft Compound Document File Format », [Internet]. Available: http://www.openoffice.org.zaxyproxy.com, p. 25, août 2007.
- [22] « Digital Corpora Downloads: corpora/files/govdocs1/threads/ ». https://downloads.digitalcorpora.org/corpora/files/govdocs1/threads/ (consulté le 7 mars 2022).