Malware Analysis

DATA ENCODING

COMPILED BY: DHARMESH DAVE | ASST. PROF. | NATIONAL FORENSIC SCIENCES UNIVERSITY

Introduction

- In the context of malware analysis, the term data encoding refers to all forms of content modification for the purpose of **hiding intent**. Malware uses encoding techniques to mask its malicious activities.
- Malware Author will choose simple ciphers or basic encoding functions that are easy to code and provide enough protection; other times, they will use sophisticated cryptographic ciphers or custom encryption to make identification and reverse-engineering more difficult.

The Goal of Analyzing Encoding Algorithms

- A malware author might use a layer of encoding for these purposes:
  - To hide configuration information, such as a command-and-control domain.
  - To save information to a staging file before stealing it.
  - To store strings used by the malware and decode them just before they are needed.
  - To disguise the malware as a legitimate tool, hiding the strings used for malicious activities.
- Our goal when analyzing encoding algorithms will always consist of two parts: **identifying the encoding functions** and then using that knowledge to **decode the attacker's secrets**.

Simple Cipher

- Simple ciphers are often disparaged for being unsophisticated, but they offer many **advantages for malware**, including the following:
  - They are small enough to be used in space-constrained environments such as exploit shellcode.
  - They are less obvious than more complex ciphers.
  - They have low overhead and thus little impact on performance.

Caesar Cipher

- One of the first ciphers ever used was the Caesar cipher. The Caesar cipher was used during the Roman Empire to hide messages transported through battlefields by courier.
- It is a simple cipher formed by shifting the letters of the alphabet three characters to the right.
- Example: key is 3
- A T T A C K AT NOON
- DWWDFN DW QRRQ

XOR Cipher

- The XOR cipher is a simple cipher that is similar to the Caesar cipher. XOR means exclusive OR and is a logical operation that can be used to modify bits.
- An XOR cipher uses a **static byte value** and modifies each byte of plaintext by performing a logical XOR operation with that **value**.
- Example with **0x3C** (hex value) **key**:

A	T	T	A	C	K		A	T		N	O	O	N
0x41	0x54	0x54	0x41	0x43	0x4B	0x20	0x41	0x54	0x20	0x4E	0x4F	0x4F	0x4E
						⇩							
}	h	h	}	DEL	W	FS	}	H	FS	r	s	s	r
0x7d	0x68	0x68	0x7d	0x7F	0x77	0x1C	0x7d	0x68	0x1C	0x72	0x71	0x71	0x72

Brute-Forcing XOR Encoding

```
5F 48 42 12 10 12 12 12 16 12 1D 12 ED ED 12 12    _HB.............
AA 12 12 12 12 12 12 12 52 12 08 12 12 12 12 12    ........R.......
12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12    ................
12 12 12 12 12 12 12 12 12 12 12 12 13 12 12 12    ................
A8 02 12 1C 0D A6 1B DF 33 AA 13 5E DF 33 82 82    ........3..^.3..
46 7A 7B 61 32 62 60 7D 75 60 73 7F 32 7F 67 61    Fz{a2b`}u`s.2.ga
```

Listing 13-1: First bytes of XOR-encoded file a.gif

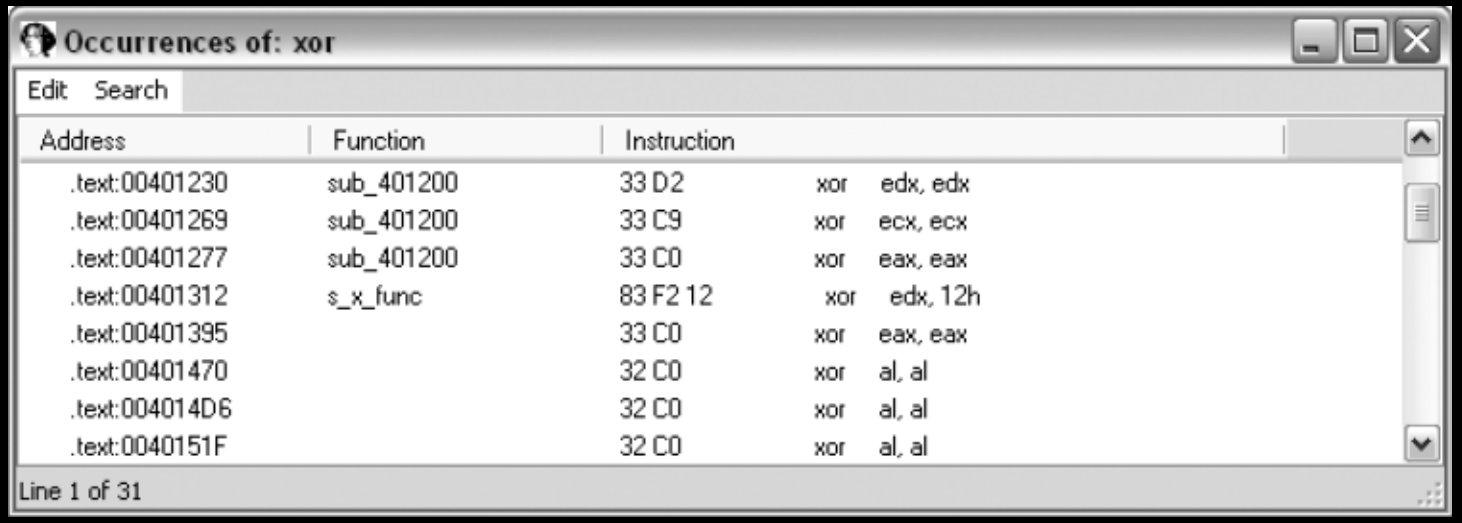
- We suspect that a.gif file may be an XOR-encoded executable because the header is not of gif file, but how do we find out? One strategy that works with single-byte encoding is brute force.
- Since there are only 256 possible values for each character in the file, it is easy and quick enough for a computer to try all of the possible 255 single byte keys XORed with the file header, and compare the output with the header you would expect for an executable file.

Brute-Forcing XOR Encoding

- The goal of brute-forcing here is to try several different values for the XOR key until you see output that you recognize—in this case, an MZ header.

XOR key value	Initial bytes of file	MZ header found?
Original	5F 48 42 12 10 12 12 12 16 12 1D 12 ED ED 12	No
XOR with 0x01	5e 49 43 13 11 13 13 13 17 13 1c 13 ec ec 13	No
XOR with 0x02	5d 4a 40 10 12 10 10 10 14 10 1f 10 ef ef 10	No
XOR with 0x03	5c 4b 41 11 13 11 11 11 15 11 1e 11 ee ee 11	No
XOR with 0x04	5b 4c 46 16 14 16 16 16 12 16 19 16 e9 e9 16	No
XOR with 0x05	5a 4d 47 17 15 17 17 17 13 17 18 17 e8 e8 17	No
...	...	No
XOR with 0x12	4d 5a 50 00 02 00 00 00 04 00 0f 00 ff ff 00	Yes!

Identifying XOR Loops with IDA Pro



- Just because a search found an XOR instruction does not mean that the XOR instruction is being used for encoding. The XOR instruction can be used for different purposes. One of the uses of XOR is to clear the contents of a register. XOR instructions can be found in three forms:
  - **XOR of a register with itself**
  - **XOR of a register (or memory reference) with a constant**
  - **XOR of one register (or memory reference) with a different register (or memory reference)**

Other Simple Encoding Schemes

Encoding Scheme	Description
ADD, SUB	Encoding algorithms can use ADD and SUB for individual bytes in a manner that is similar to XOR. ADD and SUB are not reversible, so they need to be used in tandem (one to encode and the other to decode).
ROL, ROR	Instructions rotate the bits within a byte right or left. Like ADD and SUB, these need to be used together since they are not reversible.
ROT	This is the original Caesar cipher. It's commonly used with either alphabetical characters (*A*–*Z* and *a*–*z*) or the 94 printable characters in standard ASCII.
Multibyte	Instead of a single byte, an algorithm might use a longer key, often 4 or 8 bytes in length. This typically uses XOR for each block for convenience.
Chained or loopback	This algorithm uses the content itself as part of the key, with various implementations. Most commonly, the original key is applied at one side of the plaintext (start or end), and the encoded output character is used as the key for the next character.

Base64

- Base64 encoding is used to represent binary data in an ASCII string format. Base64 encoding is commonly found in malware, so you'll need to know how to recognize it.
- The term Base64 is taken from the Multipurpose Internet Mail Extensions **(MIME) standard**. While originally developed to encode email attachments for transmission, it is now widely used for HTTP and XML.
- Base64 encoding converts **binary data into a limited character set** of 64 characters.

Base64

- Base64-encoded data **ends up being longer than the original data**. For every 3 bytes of binary data, there are at least 4 bytes of Base64-encoded data.
- The process of translating raw data to Base64 is fairly standard. It uses 24-bit (3-byte) chunks. Bits are read in **blocks of six**, starting with the most significant.
- The number represented by the 6 bits is used as an index into a 64-byte long string with each of the allowed bytes in the Base64 scheme.

A		T		T	
0x4	0x1	0x5	0x4	0x5	0x4
0 1 0 0	0 0 0 1	0 1 0 1	0 1 0 0	0 1 0 1	0 1 0 0
16	21	17	20		
Q	V	R	U		

Base64

The BASE64 Alphabet									
Char.	Dec.	Hex.		Char.	Dec.	Hex.		Char.	Dec.	Hex.
A	0	00		W	22	16		s	44	2C
B	1	01		X	23	17		t	45	2D
C	2	02		Y	24	18		u	46	2E
D	3	03		Z	25	19		v	47	2F
E	4	04		a	26	1A		w	48	30
F	5	05		b	27	1B		x	49	31
G	6	06		c	28	1C		y	50	32
H	7	07		d	29	1D		z	51	33
I	8	08		e	30	1E		0	52	34
J	9	09		f	31	1F		1	53	35
K	10	0A		g	32	20		2	54	36
L	11	0B		h	33	21		3	55	37
M	12	0C		i	34	22		4	56	38
N	13	0D		j	35	23		5	57	39
O	14	0E		k	36	24		6	58	3A
P	15	0F		l	37	25		7	59	3B
Q	16	10		m	38	26		8	60	3C
R	17	11		n	39	27		9	61	3D
S	18	12		o	40	28		+	62	3E
T	19	13		p	41	29		/	63	3F
U	20	14		q	42	2A				
V	21	15		r	43	2B		=	(pad)	(pad)

BASE64 characters are 6 bits in length. They are formed by taking a block of three octets to form a 24-bit string, which is converted into four BASE64 characters.

NOTE: The pad character (=) does not have a binary representation in BASE64; it is inserted into the BASE64 text as a placeholder to maintain 24-bit alignment.

When converting to binary, remember to use only 6 bits (e.g., 0x19 = binary 01 1001).

References

- Practical Malware Analysis by Michael Sikorski