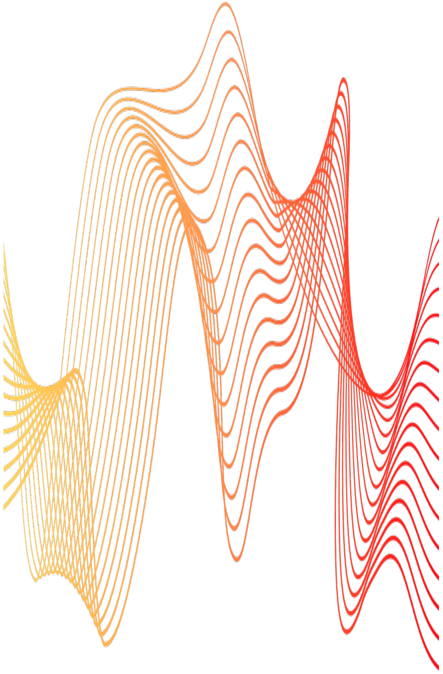
Static Analysis of Malware



Presented By-

Md. Rakibul Islam

BSSE-1411

IIT, DU

Supervisor-

Dr. B M Mainul Hossain

Professor

IIT, DU

What’s the projects all about?

Malware is a software that is specifically designed to damage computer system or take unauthorized access to a computer system. In order to protect a computer from infection or remove malware from a computer system, it is essential to accurately detect malware. So, I have decided to build up a static analyzer of malware detecting. Accordingly, the project will take a file as input and check whether the file is containing malware or not. Every file is going to be scanned, there will be generated a hash first. Once that hash is generated, the malware detector will try to match hash with its own file that file should contain hashes of known malicious files. If there is found a match, then this will be sure that the file contains malware.

Projects Motivation :-

Static analysis of malware is essential because it enables the early detection and prevention of malicious software threats, reducing the potential for harm and data breaches. It offers resource-efficient methods that are particularly vital in resource-constrained environments, ensuring robust cybersecurity measures. Additionally, static analysis contributes to improving the accuracy of antivirus solutions, enhancing their effectiveness in identifying and mitigating malware.

Key Algorithms :-

1. MD5 (Message-Digest Method 5) Hashing Algorithm

2. Fuzzy Hashing Algorithm

Progress so far :-

1.Knowing and studying about my project topic and main algorithms going to used in my project.

2.Knowing MD5 algorithm in details

3.Implementation of MD5 algorithm that finds the hash value of any input strings.

MD5 Hashing Algorithm Details:-

Introduction to MD5 Algorithm-

MD5 message-digest algorithm is the 5th version of the Message-Digest Algorithm developed by Ron Rivest to produce a 128-bit message digest. MD5 is quite fast than other versions of the message digest, which takes the plain text of 512-bit blocks, which is further divided into 16 blocks, each of 32 bit and produces the 128-bit message digest, which is a set of four blocks, each of 32 bits. MD5 produces the message digest through five steps, i.e. padding, append length, dividing the input into 512-bit blocks, initialising chaining variables a process blocks and 4 rounds, and using different constant it in each iteration.

Five steps of MD5 Hashing Algorithm :-

1. Append Padding Bits
2. Append Length
3. Initialize MD Buffer
4. Process Message in 16-Word Blocks
5. Output

Step-1

Appending Padding Bits. The original message is "padded" (extended) so that its length (in bits) is congruent to 448, modulo 512. The padding rules are:

1. The original message is always padded with one bit "1" first.
2. Then zero or more bits "0" are padded to bring the length of the message up to 64 bits fewer than a multiple of 512.

Step-2

Appending Length. 64 bits are appended to the end of the padded message to indicate the length of the original message in bytes. The rules of appending length are:

1. The length of the original message in bytes is converted to its binary format of 64 bits. If overflow happens, only the low-order 64 bits are used.
2. Break the 64-bit length into 2 words (32 bits each).

The low-order word is appended first and followed by the high-order word.

Step-3

Initializing MD Buffer. MD5 algorithm requires a 128-bit buffer with a specific initial value. The rules of initializing buffer are:

The buffer is divided into 4 words (32 bits each), named as A, B, C, and D.

Word A is initialized to: 0x67452301.

Word B is initialized to: 0xEFCDAB89.

Word C is initialized to: 0x98BADCFE.

Word D is initialized to: 0x10325476.

Step-4

Processing Message in 512-bit Blocks. This is the main step of MD 5 algorithm, which loops through the padded and appended message in blocks of 512 bits each. For each input block, 4 rounds of operations are performed with 16 operations in each round.

Input and predefined functions:

A, B, C, D: initialized buffer words

F(X,Y,Z) = (X AND Y) OR (NOT X AND Z)
G(X,Y,Z) = (X AND Z) OR (Y AND NOT Z)
H(X,Y,Z) = X XOR Y XOR Z
I(X,Y,Z) = Y XOR (X OR NOT Z)

T[1, 2, ..., 64]: Array of special constants (32-bit integers) as:

T[i] = int(abs(sin(i)) * 2**32)

M[1, 2, ..., N]: Blocks of the padded and appended message

R1(a,b,c,d,X,s,i): Round 1 operation defined as:

$$a = b + ((a + F(b,c,d) + X + T[i]) <<< s)$$

R2(a,b,c,d,X,s,i): Round 1 operation defined as:

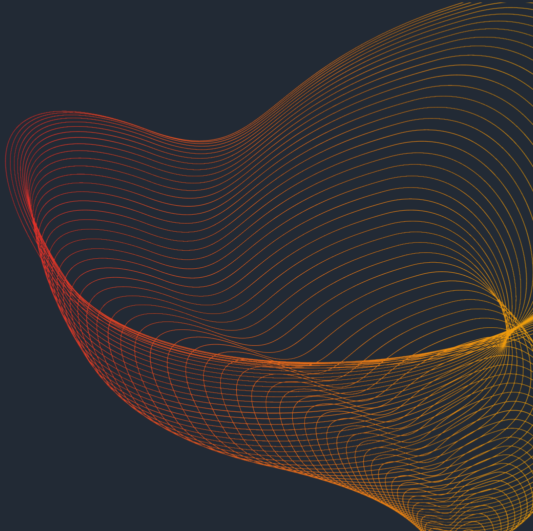
$$a = b + ((a + G(b,c,d) + X + T[i]) <<< s)$$

R3(a,b,c,d,X,s,i): Round 1 operation defined as:

$$a = b + ((a + H(b,c,d) + X + T[i]) <<< s)$$

R4(a,b,c,d,X,s,i): Round 1 operation defined as:

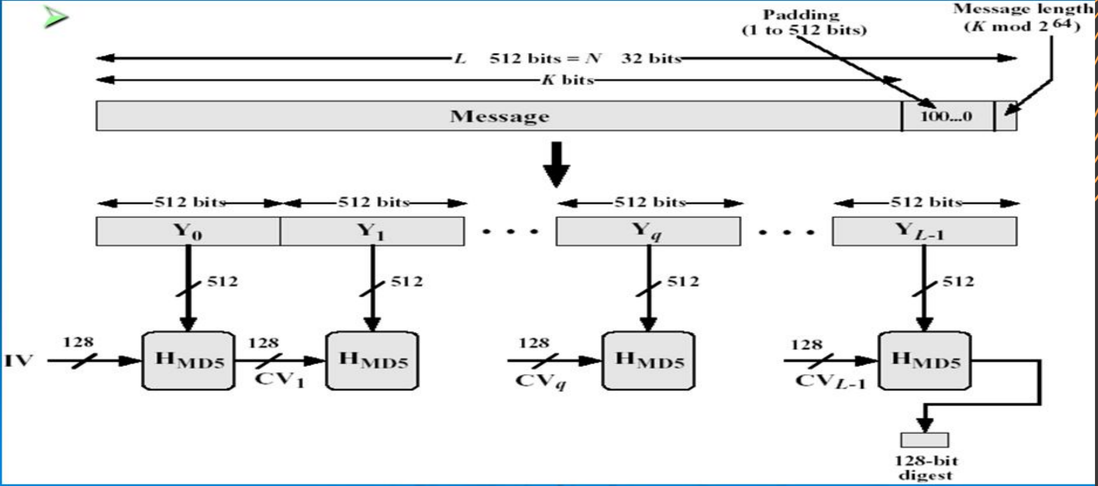
$$a = b + ((a + I(b,c,d) + X + T[i]) <<< s)$$



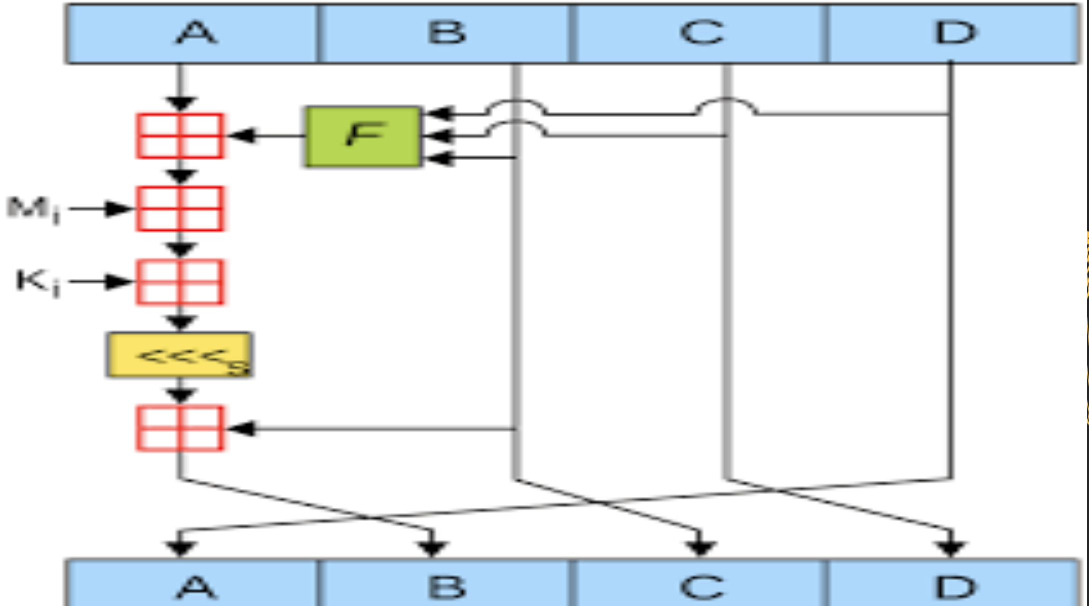
Step-5

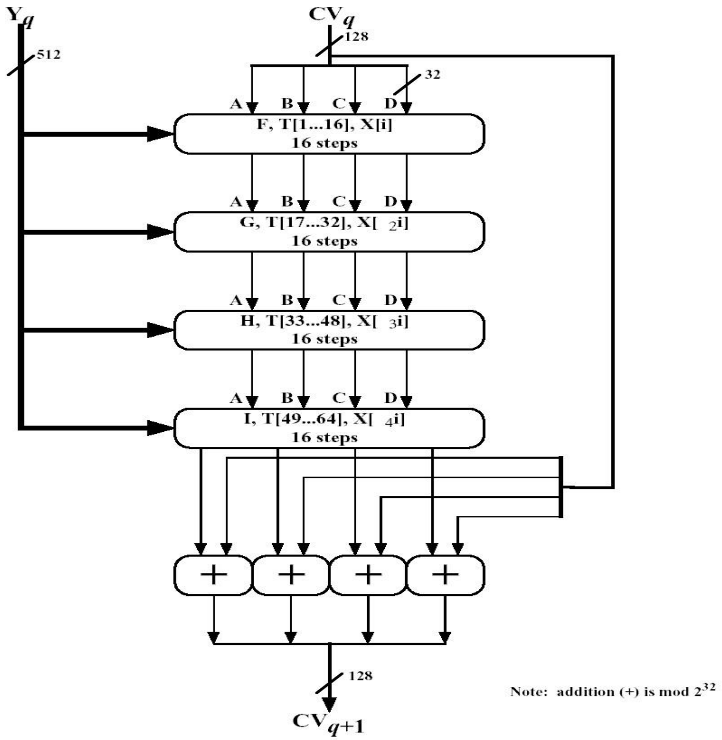
Output. The contents in buffer words A, B, C, D are returned in sequence with low-order byte first.

MD5 Algorithm Architecture



MD5 Algorithm One MD5 Operation :-

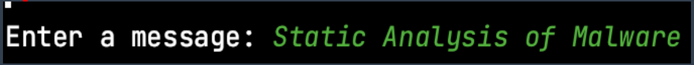




Y_q
CV_q
512
128
32
A
B
C
D
F, T[1...16], X[i]
16 steps
G, T[17...32], X[ρ_2i]
16 steps
H, T[33...48], X[ρ_3i]
16 steps
I, T[49...64], X[ρ_4i]
16 steps
+
128
CV_{q+1}
Note: addition (+) is mod 2^{32}

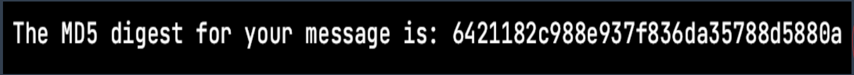
Figure 9.2 MD5 Processing of a Single 512-bit Block (MD5 Compression Function)

input :-



```
Enter a message: Static Analysis of Malware
```

Output :-



```
The MD5 digest for your message is: 6421182c988e937f836da35788d5880a
```

Future Works :-

1.Now I read a text file, any strings as input and generate its hash value, next I will read a exe file as input and find it's hash value and store a file and compare its hash value with knowing malicious file hash value.

2.Then i will apply Fazzy hashing for comparing exe file similarity and store the hash value in a file.

Tools Used:-

1.Language: C/C++

2.Github link: https://github.com/Rakibul1411/SPL-01-3rd-Semester-

Thank you