Seguridad en un Sistema Operativo: Malware On Disk implementando un Ransomware de Encriptación

Esteban Alvarado Vargas*, Olman Castro Hernández†, Martín Calderón Blanco‡, María Lucía Monge Golcher§
Área Académica de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica
Email: * estalvgs1999@estudiantec.cr, † olcastro@estudiantec.cr, ‡ martinrolo22@estudiantec.cr, § luciamongeg@gmail.com

***Abstract*—In this paper, the main topic is security at the operating system. The Vigenere encryption algorithm, which is based on polyalphabetic substitution, will be used to encrypt a file and generated key that will be decrypted with the help of brute force, this algorithm checks all possible combinations until it finds the correct key. In this way the resources and time consumed will be measured until the process is complete in order to analyze the obtained results. These have a great impact, even consuming 100 porcent of the CPU in some cases, so the conclusion is reached that malware is a really dangerous software which can take over the system and the data in it, also the big importance of not exposing yourself to possible cyberattacks with the help of encryption algorithms.**

***Palabras clave*—Algoritmos, Ciberseguridad, Encriptado, Malware, Ransomware.**

I. Introducción

La ciberseguridad o seguridad informática es un área de gran importancia en un mundo donde la tecnología avanza a grandes pasos y la información crítica o no, es almacenada y debe ser protegida de posibles ultrajes que atenten contra las personas y su integridad. De esta forma, los programadores tienen el deber moral y ético de crear sistemas seguros que cuenten con la encriptación de la información necesaria para asegurar casi en su totalidad un sistema robusto y que pueda soportar los posibles saltos de parte de personas malintencionadas.

Este proyecto aborda la seguridad de un sistema operativo cuando este se encuentra bajo el ataque de un programa malicioso, específicamente se busca demostrar de forma práctica cuando un archivo es encriptado con un algoritmo para cifrar su información y se le quiere emplear el conocido algoritmo de fuerza bruta para con esto verificar tanto el consumo de recursos como el tiempo en disco, de forma que con los datos obtenidos se pueda comprobar la utilidad de la encriptación aplicada al archivo de prueba. Se espera que al aplicar el algoritmo de encriptado al archivo, este genere un código de cifrado tan complejo que el algoritmo de decodificación por fuerza bruta consuma una gran cantidad de recursos y tarde un tiempo considerable para descifrar dicho código.

II. Background

Inicialmente, un malware (*software malicioso*) es un programa que ingresa a un sistema sin el consentimiento del usuario, este tiene la intención de robar información personal o datos confidenciales como cuentas de bancos o contraseñas. Adicionalmente, puede ralentizar el computador, mostrar continuamente anuncios emergentes e incluso modificar la configuración del sistema. [1] Hay diferentes formas en que un malware puede ingresar al sistema, a continuación se mencionan algunas:

- Descargar un programa gratuito o archivos, en una página no segura.
- Al abrir correos sospechosos.
- Ingresar a sitios web maliciosos.
- Insertar dispositivos de almacenamiento infectados al computador.

De estas formas, el malware entra al equipo y lo infecta ”por debajo de la mesa”. Existen varios tipos de software malicioso, por ejemplo:

- **Virus.** Este es un programa que daña el sistema computacional o elimina información, su propósito es infectar sistemas vulnerables para ganar el control de administrador y borra o robar datos. [1]
- **Worm.** Es un programa que busca llenar el sistema, replicándose constantemente y disminuyendo la velocidad de cómputo.[1]
- **Logic Bomb.** Por otro lado, el *Logic Bomb* es ejecutado cuando cierta acción ocurre y se mantienen ocultos.[1]
- **Trojan o Backdoor.** Estos pretenden ser programas inofensivos como juegos o aplicaciones, al ser ejecutados se dispara generando daños, usualmente contienen herramientas para monitorear la actividad del usuario.[1]

Estos programas maliciosos nos llevan a querer proteger nuestra información, por lo que se han desarrollado técnicas para proteger los datos de las personas y entre estas se encuentran los algoritmos de encriptación. Sin embargo, estos algoritmos también se han aprovechado para provocar daños en los sistemas.

Existe un tipo de malware llamado *Ransomware*, que es una extorsión por dinero por parte de un atacante en la que los archivos del usuario se encriptan y la clave de descifrado es retenida por los atacantes, hasta que se recibe una cantidad de rescate de la víctima [2]. Se realizó una investigación con el fin de conocer a profundidad el funcionamiento y propósito

de varios de estos algoritmos de encriptación. Algunos de los investigados se comentan a continuación:

- *Secure Hash Algorithm (SHA):* Es una función hash criptográfica ampliamente utilizada, que genera un hash de 160 bits (20 bytes) a partir de cualquier valor de entrada. Esto se utiliza para calcular un valor de comprobación única para todos los datos digitales de no más de 264 -1 bit de longitud y es la base para la creación de una firma digital. [3]
- *RSA*: Es un algoritmo de cifrado asimétrico o de clave pública, es uno de los más utilizados en la actualidad. Se generan las llaves, y se cifra el texto por medio de ecuaciones matemáticas y la llave pública. Con la llave privada se descifra el texto. [3]
- *Argon2*: Tiene dos versiones distintas: Argon2d y Argon2i, el primero depende de los datos y el segundo es independiente de estos. El primero es resistente al craqueo de la GPU, mientras que el segundo es resistente a los ataques de canal lateral. En otras palabras, Argon2d sería adecuado para el hash de contraseñas, mientras que Argon2i sería adecuado para la derivación de claves de cifrado. [3]
- *Scrypt*: Realiza un hash utilizando una clave, una serie de puntos marcados en el algoritmo hash y agregando mucho ruido, el cual es una serie de números aleatorios generados por el algoritmo y almacenados en memoria. El fin de estos números es camuflar los datos claves del algoritmo, para hacer más complejo el trabajo de romper dichos hash. [3]
- *Vigenere Cipher*: Este cifrado es de sustitución, es utilizado para encriptar datos en el que el formato del texto está oculto en el texto cifrado mediante el uso de varios cifrados mono-alfabéticos diferentes en lugar de uno solo. El código especifica qué sustitución en particular se empleará para cifrar cada símbolo de texto sin formato. Los sistemas difieren principalmente en la forma en que se utiliza la clave para elegir entre la colección de reglas de sustitución mono-alfabéticas. [4] En los sistemas más simples la clave es una palabra o frase que se repite tantas veces como sea necesario para cifrar un mensaje.

Este último es el seleccionado y desarrollado en el proyecto, por lo cual será desarrollado en mayor medida en la implementación. Por otra parte, tenemos el algoritmo de descifrado por fuerza bruta, del cual busca la combinación necesaria hasta encontrarla pasando por todas las soluciones posibles, en el mejor de los casos esta se encuentra entre las primeras combinaciones generadas, en el peor, debe intentar todas las combinaciones posibles hasta que la última es la llave correcta. Algunas ventajas y desventajas son las siguientes:

- **Ventajas**
  a. Garantiza encontrar la solución correcta.
  b. Es un método genérico que no está limitado.
  c. Es ideal para resolver problemas simples y pequeños.
- **Desventajas**
  a. Es muy ineficiente, puede llegar al orden de crecimiento O(N!).
  b. Es lento para encontrar la solución.
  c. Tiende a comprometer el poder del sistema computacional.

III. Implementación

Esta sección del paper brinda una descripción del experimento propuesto. Se consideraron distintos algoritmos de encriptación para realizar el cifrado, pero, finalmente, se seleccionó el cifrado Vigenere. Como se mencionó anteriormente, el cifrado Vigenere se basa en la sustitución poli alfabética, con una llave igual al tamaño del contenido.

Hemos desarrollado un programa computacional que permite encriptar un archivo de texto con el cifrado Vigenere, utilizando una llave generada de manera aleatoria, para luego ejecutar un algoritmo de descifrado a fuerza bruta para recuperar el contenido original del archivo.

A. Cifrado Vigenere

El cifrado Vigenere es un cifrado basado en la sustitución poli alfabética. Se conocía en el siglo XIX como el código indescifrable, pues aunque es sencillo de implementar, parece imposible de resolver. Este algoritmo utiliza un conjunto de caracteres que conforman un alfabeto. Con este conjunto se confecciona una tabla conocida como la tabla de Vigenere, que contiene en la primera fila todo el alfabeto en orden, a partir de la segunda fila, cada caracter se desplaza a la izquierda una posición [5]. La Figura 1 presenta la tabla de Vigenere para el alfabeto de letras mayúsculas en inglés.

Fig. 1: Tabla de Vigenere para el alfabeto en Inglés (26 caracteres)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

El algoritmo de cifrado incluye los siguientes pasos [5]:

1) Se debe definir el conjunto de caracteres que conforman el alfabeto.
2) Determine la llave de cifrado.
3) Ajuste la llave de cifrado al largo del contenido, repitiendo la clave la cantidad de veces que sea necesario.
4) Se debe recorrer el contenido. Para calcular el caracter encriptado se aplica la siguiente ecuación: $E_k(C_i) = (C_i - K_i) \,\%\, N$, donde C_i es el caracter del

contenido, K_i es el caracter de la llave para esa posición y N es el tamaño del alfabeto.

5) Al terminar de realizar la iteración, el resultado es el contenido encriptado.

El Algoritmo 1 presenta el pseudocódigo del programa que implementa el cifrado. Y el Algoritmo 2 muestra la secuencia de encriptación de un archivo.

Algorithm 1: Algoritmo de Vigenere para el cifrado de un archivo utilizando una llave de encriptación

```
function VigenereCipher (content, key);
  Input : El contenido content y la llave de cifrado key
  Output: (out, key)
for i in len(content) do
    indexContent ← ord(content[i]) - min;
    indexKey ← ord(key[i]) - min;
    diff ← indexContent - indexKey;
    indexCipher ← -fmod(diff, (max - min+1));
    out ← out + alphabet[indexCipher];
end
return out, key;
```

Algorithm 2: Algoritmo de encriptación utilizando el cifrado de Vigenere

```
function Encrypt (filename, length);
  Input : El nombre del archivo filename y el largo de la llave de cifrado length
content ← read file;
copy file to fileBackup;
key ← GenerateKey (length);
content ← VigenereCipher (content,key);
write content → file;
```

B. Ataque de Fuerza Bruta

El *ataque a fuerza bruta* para desencriptar el contenido, se realiza probando todas las posibles llaves hasta encontrar la que recupera el contenido. En el Algoritmo 3 se ilustra con un pseudocódigo, el programa implementado. En este, primero se generan todas las combinaciones de una llave de tamaño L y luego se prueba cada una, descifrando el código con la llave de turno y comparando el contenido original con el descifrado. Si ambos coinciden se escribe el contenido recuperado en el archivo y finaliza la ejecución, si no, se continúan probando llaves hasta que la encuentre.

La cantidad de posibles combinaciones que puede tener una llave se calcula en función de la longitud de la misma. Para un alfabeto de clave de largo L, la ecuación es: $P(n) = L^n$. La Tabla I, muestra la cantidad de combinaciones posibles en función del tamaño, para una llave compuesta por letras mayúsculas del alfabeto en inglés.

Entre más larga sea la llave, se vuelve mucho más complicado descifrarla por fuerza bruta, pues un caracter adicional implica un incremento exponencial en las posibilidades y esto aumenta el tiempo de cómputo y la exigencia de recursos. Esto nos indica que este algoritmo puede considerarse bastante seguro.

Algorithm 3: Algoritmo de recuperación por fuerza bruta

```
function DecryptBruteForce (filename, length);
  Input : El nombre del archivo cifrado filename y el largo de la llave de cifrado length
EncryptedContent ← read file;
OriginalContent ← read fileBackup;
for key in GenerateKeyCombinations do
    DecryptedContent ← VigenereCipher (EncryptedContent,key);
    if DecryptedContent == OriginalContent then
        write DecryptedContent → file;
        break;
    end
end
```

Tamaño de Llave	Ecuación	Llaves posibles
1	26^1	26
2	26^2	676
3	26^3	17576
4	26^4	456976
5	26^5	11881376
6	26^6	308915776
7	26^7	8031810176
8	26^8	208827064576
9	26^9	5429503678976
10	26^10	141167095653376
11	26^11	3670344486987776
12	26^12	95428956661682176
13	26^13	2481152873203736576
14	26^14	64509974703297150976

TABLE I: Cantidad de posibles combinaciones que puede tener una llave para un alfabeto de 26 elementos

C. Experimento

El experimento consiste en:

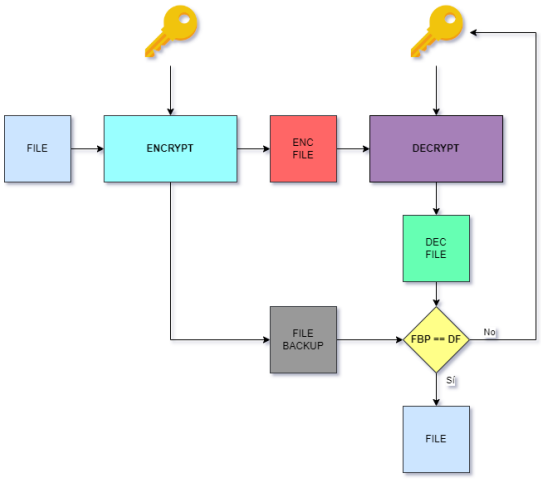
- Encriptar un archivo de texto con una llave aleatoria y desconocida.
- Desencriptar el archivo utilizando un ataque a fuerza bruta.
- Medir el tiempo que tarda el algoritmo en encontrar la llave de cifrado.
- Medir el consumo de recursos que utilizó el programa para encontrar la llave.
- Repetir para distintos tamaños de llave: 1, 2, 3, 4, 5 y 6.

La Figura 2 ilustra los pasos del experimento en un diagrama de flujo.

D. Prevención

Ya que una de las formas de infectar el sistema con un Ransomware es a través de sitios web no seguros, este problema puede verse prevenido al no acceder a estos sitios, además de no entrar a enlaces desconocidos o que tienen su

Fig. 2: Diagrama de flujo del experimento



enlace reducido con el fin de esconder su dirección completa, asimismo siempre que un enlace está compuesto por números o caracteres que simulen un sitio conocido este debe evitarse. Además de evitar los enlaces anteriores, también es importante identificar correos electrónicos sospechosos y si alguna duda surge con respecto a este, lo mejor sería marcar como correo basura, eliminar el mensaje o bloquear al emisor.

Por otro lado, también se pueden transmitir de por medio de memorias USB, por lo que es recomendado que no conectar estos dispositivos si no se conoce su contenido, igualmente al mantener los programas y el sistema operativo actualizado se disminuye el riesgo a sufrir por un programa malicioso. Por último, en el caso en que el sistema se vea afectado o comprometido, mantener copias de seguridad de información valiosa es de gran importancia, ya que aunque el sistema tenga que pasar por un proceso para ser recuperado en el que posiblemente la información deba ser borrada, se tendrán los datos de mayor importancia en lugares seguros como por ejemplo discos externos.

IV. Resultados y Análisis

El experimento se llevó a cabo en dos equipos con Linux. Primero, en una máquina limitada, con un procesador de 1 núcleo y 1 GB de RAM con el sistema operativo CentOS y luego, en un equipo con un procesador de 4 núcleos y 8 GB de RAM con sistema operativo Linux Mint.

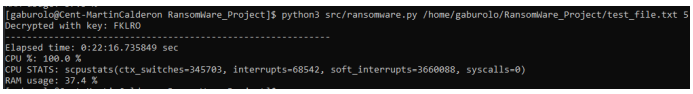
Para el experimento se usó un archivo de texto cuyo contenido está conformado por caracteres ASCII y un generador de llaves de 26 caracteres, se emplean las letras mayúsculas del alfabeto en inglés.

A. Ejecución en CentOS

En la máquina de bajo rendimiento se llevaron a cabo tres ejecuciones, para largos de llave de 3, 4 y 5 caracteres. La Figura 3 muestra una captura de pantalla tomada durante la ejecución de uno de los experimentos.

Los resultados obtenidos de cada experimento se presentan en las Tablas II, III y IV. En todos los resultados se puede

Fig. 3: Captura de pantalla de la ejecución del experimento en CentOS.



observar con claridad que el programa exige completamente el procesador, utilizando el 100% de CPU. En el consumo de memoria se obtiene una utilización promedio del 30.86%.

Parámetros/Llave	DNL
Tiempo	0:00:01.272536
CPU %	100,00%
RAM %	37,20%

TABLE II: Tiempo de ejecución y consumo de recursos de la ejecución con llave de largo 3 en CentOS

Parámetros/Llave	WFWC
Tiempo	0:03:30.606451
CPU %	100,00%
RAM %	37,30%

TABLE III: Tiempo de ejecución y consumo de recursos de la ejecución con llave de largo 4 en CentOS

Parámetros/Llave	FKLRO
Tiempo	0:22:16.735849
CPU %	31,00%
RAM %	18,20%

TABLE IV: Tiempo de ejecución y consumo de recursos de la ejecución con llave de largo 5 en CentOS

El rendimiento en tiempos de ejecución demuestra que el tiempo que tarda el algoritmo en encontrar la llave correcta está relacionado con qué tan cercana esté alfabéticamente. De esta forma, la variable es la longitud de la llave, pues, como se evidenció anteriormente, entre más larga la llave, mayor es el número de combinaciones posibles.

B. Ejecución en Linux Mint

Esta sección presenta los resultados de los experimentos realizados en la máquina de mayor capacidad. Dieciséis ejecuciones se llevaron a cabo, tres longitudes de llave de 1 a 5 y una para la llave de longitud 6. La Figura 4 muestra una captura de pantalla tomada durante la ejecución del experimento para una llave de 6 caracteres.

Los resultados obtenidos por cada experimento se han agrupado en tablas para una mejor visualización de los datos.

En los resultados se observa un consumo de recursos relativamente constante para las diferentes longitudes de llave, el porcentaje de procesador promedio se encuentra en 31.33% y la utilización de memoria RAM se encuentre en el rango del 28%.

El tiempo de ejecución se puede analizar más a profundidad gracias a que hay una mayor cantidad de datos. De los tiempos

Parámetros/Llave	F	R	O
Tiempo	0:00:00.003975	0:00:00.007055	0:00:00.008856
CPU %	33,30%	25,00%	33,30%
RAM %	29,20%	28,40%	28,90%

TABLE V: Tiempo de ejecución y consumo de recursos de la ejecución con llave de largo 1 en Linux Mint

Parámetros/Llave	TR	VP	WS
Tiempo	0:00:00.254676	0:00:00.262339	0:00:00.278530
CPU %	33,50%	36,10%	33,30%
RAM %	29,40%	28,80%	28,40%

TABLE VI: Tiempo de ejecución y consumo de recursos de la ejecución con llave de largo 2 en Linux Mint

Parámetros/Llave	TIY	MBJ	DEC
Tiempo	0:00:05.328360	0:00:03.394387	0:00:00.939493
CPU %	33,10%	34,60%	36,30%
RAM %	29,20%	28,70%	28,50%

TABLE VII: Tiempo de ejecución y consumo de recursos de la ejecución con llave de largo 3 en Linux Mint

Parámetros/Llave	CWTW	LXVE	XAUV
Tiempo	0:00:20.566882	0:01:28.071411	0:02:43.536531
CPU %	35,10%	36,00%	33,10%
RAM %	28,30%	28,40%	28,10%

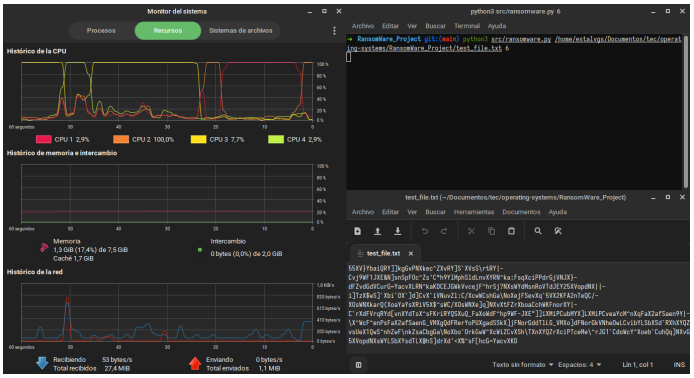
TABLE VIII: Tiempo de ejecución y consumo de recursos de la ejecución con llave de largo 4 en Linux Mint

Parámetros/Llave	DMLQI	XYHGU	TJWGD
Tiempo	0:09:44.670260	1:13:35.004681	0:59:47.885424
CPU %	30,00%	32,50%	32,50%
RAM %	28,70%	28,50%	28,20%

TABLE IX: Tiempo de ejecución y consumo de recursos de la ejecución con llave de largo 5 en Linux Mint

obtenidos en la Tabla V al tiempo medido en la Tabla X se puede notar un incremento considerable. Sin embargo, los resultados obtenidos corresponden a los tiempos que tardó el programa en encontrar la llave aleatoria, por lo que no se obtuvo una medida objetiva del peor caso posible.

Fig. 4: Captura de pantalla de la ejecución del experimento en Linux Mint.

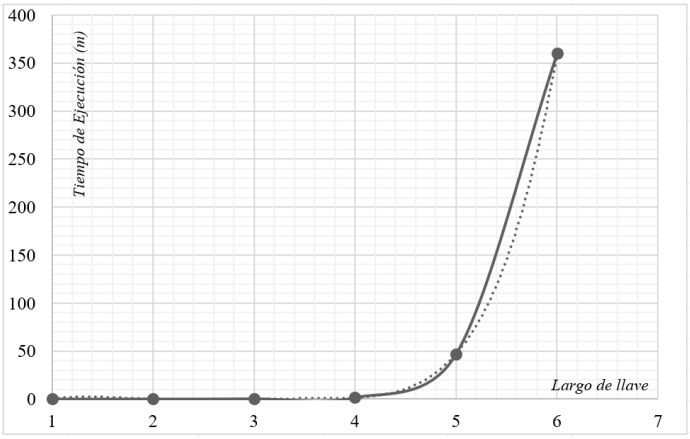


Parámetros/Llave	FHGELL
Tiempo	6:52:54.153452
CPU %	31,00%
RAM %	18,20%

TABLE X: Tiempo de ejecución y consumo de recursos de la ejecución con llave de largo 6 en Linux Mint

Para realizar este análisis, se confeccionó la Tabla XI en la que se calcula una aproximación del tiempo que tardaría en ejecutar el peor caso posible, es decir, cuando la llave corresponde a la combinación 26^n. Además, se confeccionó el gráfico de la Figura 5, que muestra el tiempo que tomó resolver el problema para cada valor de longitud.

Fig. 5: Gráfico del tiempo de ejecución del ataque de fuerza bruta en función del largo de la llave.



Se puede observar que el comportamiento del programa es exponencial y proporcional al largo de la llave. Este resultado concuerda con la teoría, ya que, entre más posibilidades existan, más tiempo le tomará al programa comprobarlas todas. Esto nos revela lo costoso que resulta descifrar este algoritmo de cifrado cuando la llave es lo suficientemente larga, llegando a tardar días o años para poder encontrarla.

Tamaño	Tiempo Medido	% Combinaciones	Peor caso (m)
1	0:00:00.008	69.23	0.02
2	0:00:00.278	87.42	0.53
3	0:00:01.000	12.14	13.73
4	0:02:43.587	88.58	307.80
5	0:59:47.885	74.58	8 018.24
6	6:52:12.602	20.20	204 064.37

TABLE XI: Tiempo de ejecución aproximado para el peor caso

V. Conclusiones

Al estresar de manera controlada los diferentes recursos computacionales se puede observar el peligro que puede significar que un malware o Ransomware descontrolado infecte nuestros sistemas, teniendo el potencial para controlar e infectar todo a su paso.

De los resultados del experimento se concluye que un Ransomware resulta especialmente peligroso, pues puede encriptar archivos críticos de un sistema y causar graves daños, y recuperarlo cuesta demasiado en tiempo y recursos.

También se encontró que existe una relación exponencial entre el tiempo de ejecución del ataque a fuerza bruta y el tamaño de la llave que se está buscando.

Un ataque por Ransomware es difícil de solucionar y la mejor defensa es tomar medidas de seguridad preventivas para minimizar las posibilidades de una infección y garantizar una posible recuperación de los datos mediante un respaldo.

VI. Sugerencias y recomendaciones

En cuanto a seguridad, es recomendable revisar y conocer más algoritmos de encriptación con el fin de analizar los diferentes resultados que puedan tener y de esta forma utilizar el algoritmo que se adecue a las posibles necesidades y recursos en que sea aplicado.

Por parte del posible riesgo, se debería tomar en cuenta los algoritmos especiales para desencriptar a la hora de realizar pruebas, los cuales pueden ser más eficientes a la hora de encontrar llaves en comparación con fuerza bruta y así tener un rango de conocimiento mayor sobre los riesgos a los que se puede exponer, además de utilizar hilos para estresar aún más el sistema y probar los peores escenarios posibles.

VII. Referencias

References

[1] G. Mohit, ”Malwares – Malicious Software”, GeeksforGeeks, 2020. [Online]. Available: https://www.geeksforgeeks.org/malwares-malicious-software/

[2] N. Aldaraani and Z. Begum, ”Understanding the impact of Ransomware: A Survey on its Evolution, Mitigation and Prevention Techniques,” 2018 21st Saudi Computer Society National Computer Conference (NCC), 2018, pp. 1-5, doi: 10.1109/NCG.2018.8593029.

[3] L. Alberto, ”Criptografía: Qué son los algoritmos hash y para qué se utilizan”, Redes Zone, 2021. [Online]. Available: https://www.redeszone.net/tutoriales/seguridad/criptografia-algoritmos-hash/

[4] S. Gustavus, ”Vigenère cipher cryptology”, Encyclopedia Britannica. [Online]. Available: https://www.britannica.com/topic/Vigenere-cipher

[5] C. K.Shene, ”Cryptography Visualization Tools”, Department of Computer Science Michigan Technological University, 2014.