Estudio práctico de técnicas de ofuscación y contramedidas aplicables

María San José Seco
@drkrysSrng/freyja

Universidad Católica de Murcia
ENIIT - Campus Internacional de Ciberseguridad

1. Introducción

A lo largo de la historia, tanto como para proteger la propiedad intelectual o intercambiar secretos, se ha utilizado la ofuscación como medida de protección. El malware la ido evolucionando a través de los años utilizando diferentes técnicas cada vez más avanzadas de manera que los antivirus y los analistas no puedan detectarlos y evadirlos, ya que este método previene los análisis por firma o hash o por las reglas YARA que ayudan a buscar cadenas de texto sospechosas dentro de ellos.

Hoy en día, el tipo de malware que más nos podemos encontrar, es metamórfico, ya que se ha avanzado mucho para evadir los antivirus. De esta manera, la evolución del malware parte desde el punto de encontrarnos muestras oligomórficas, donde la parte viral está encriptada, muestras polimórficas, donde no sólo está encriptado sino que también está ofuscado. El primer malware polimórfico de la Historia, Luna fue desarrollado en España por Bumblebee en el año 1999.

Analizar la Entropía de una muestra es importante, de esa manera, se puede identificar la aleatoriedad o desorden de un sistema, de manera que podamos identificar si una muestra ha sido ofuscada o no. Cuanto más alta sea la probabilidad y sobre todo mayor de *3.75* significa que no ha sido escrito por un humano.

2. Importancia de las amenazas de JavaScript en Windows

Los ataques basados en script se han convertido en una amenaza importante en los últimos años. Siendo el 40 % de los últimos años, entre los lenguajes más utilizados se encuentran PowerShell, VBScript y JavaScript. Sin embargo la mayoría de malware se están migrando a éste último ya que es un lenguaje que se puede ejecutar dentro y fuera del navegador en sistemas Windows.

3. Freyja Deobfuscation Tool

@drkrysSrng/freyja

Se ha desarrollado la siguiente herramienta para aplicar varios tipos de ofuscación mencionados en el apartado anterior:

- Limpia el código y lo tabula de manera similar, no sólo tabulando las instrucciones, sino que parsea los caracteres en hexadecimal y unicode.
- Chequeeo de la entropía con el algoritmo de Shannon, línea por línea o por texto completo.
- Cuando los caracteres Unicode o en Hexadecimal no están parseados ya que no están en formato String sino que es una variable ofuscada, se parsean a mayores también.
- Desofusca funciones como `toString`, conjuntos de números dentro de `eval`, sustituir `unescape` y `parseint`:
- Concatena cadenas de caracteres separadas con el símbolo `+`
- Búsqueda de strings en base64 y su decodificación.

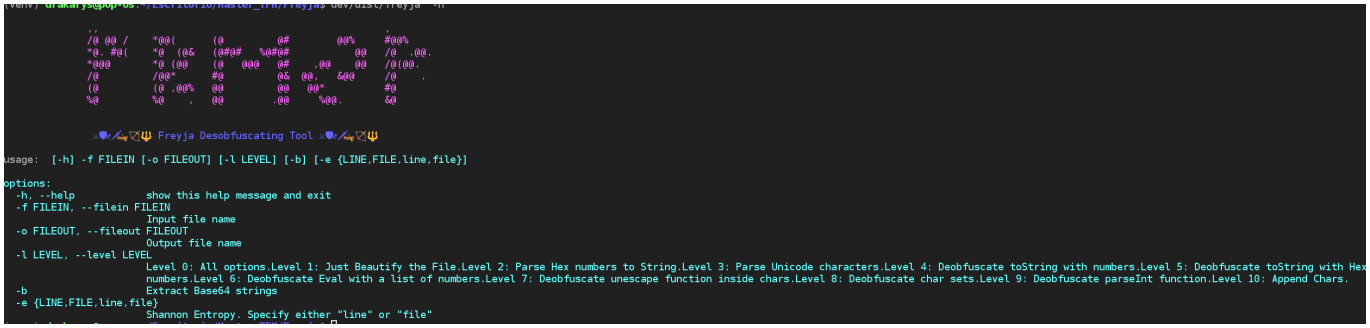

Figura 1: Uso de la herramienta Freyja

- `-f` Fichero a desofuscar.
- `-o` Fichero de salida desofuscado.
- `-l` Nivel de ofuscación. Le indicamos la técnica que queremos que utilice.
  - Nivel 0: Todas las opciones
  - Nivel 1: Técnica *beautify*, tabular el fichero y darle formato al JavaScript.
  - Nivel 3: Parsear los caracteres Unicode a Strings.
  - Nivel 4: Desofuscar la función `toString`.
  - Nivel 5: Desofuscar la función `toString` con número hexadecimal.
  - Nivel 6: Desofuscar la función `eval` cuando tiene una lista de números.
  - Nivel 7: Desofuscar la función `unescape` cuando contiene caracteres.
  - Nivel 8: Desofuscar conjuntos de caracteres.
  - Nivel 9: Desofuscar la función `parseInt`.
  - Nivel 10: Concatena caracteres aunque estén en varias líneas.