Antimalware Scan Interface (AMSI)

Enumerables de amsi.h

amsi.h utiliza el enumerable **`AMSI_RESULT`** como tipo de respuesta para el scanbuffer, en el caso de c# scanbuffer devuelve un int.

```
typedef enum AMSI_RESULT {
  AMSI_RESULT_CLEAN,
  AMSI_RESULT_NOT_DETECTED,
  AMSI_RESULT_BLOCKED_BY_ADMIN_START,
  AMSI_RESULT_BLOCKED_BY_ADMIN_END,
  AMSI_RESULT_DETECTED
} ;
```

`AMSI_RESULT_CLEAN:` Bien conocido. No se encontró ninguna detección y es probable que el resultado no cambie después de una actualización de definición futura.

`AMSI_RESULT_NOT_DETECTED:` No se encontró ninguna detección, pero el resultado podría cambiar después de una actualización de definición futura.

`AMSI_RESULT_BLOCKED_BY_ADMIN_START:` La directiva de administrador bloqueó este contenido en esta máquina (principio del intervalo).

`AMSI_RESULT_BLOCKED_BY_ADMIN_END:` La directiva de administrador bloqueó este contenido en esta máquina (final del intervalo).

`AMSI_RESULT_DETECTED:` Detección encontrada. El contenido se considera malware y debe bloquearse.

En nuestra aplicacion de C# al escanear el buffer con la **.dll** de amsi.h el mismo devuelve un numero entero, luego no tenemos directamente el enum proporcionado por **amsi.h** en **c++**, en el caso de C# entonces, definimos intervalos de enteros que definiran que tipo de respuesta devuelve el **ScanBuffer**, estos intervalos de enteros son redefinibles desde el codigo, en nuestra aplicacion utilizamos los datos proporcionados por la pagina oficial de windows:
https://learn.microsoft.com/es-es/windows/win32/api/amsi/ne-amsi-amsi_result

Entonces dentro de la funcion `HasMalware` se definen los siguientes enteros:

```
const int AMSI_RESULT_CLEAN = 0,
          AMSI_RESULT_NOT_DETECTED = 1,
          AMSI_RESULT_BLOCKED_BY_ADMIN_START = 16384,
          AMSI_RESULT_BLOCKED_BY_ADMIN_END = 20479,
          AMSI_RESULT_DETECTED = 32768;
```

Esta funcion tiene como entrada un entero el cual analizara y tomara accion segun su rango, si la entrada es mayor que 32768 entonces la considerara malware y la funcion devuelve **`true`**, si la entrada esta en el intervalo (`16384,20479`) lanza una exepcion diciendo que la politica de administracion ha bloqueado este archivo para esta maquina. En otro caso decide que no es un malware y devuelve **`false`**.
Mientras mayor sea el numero numero que se analiza mas peligrosidad representa. Entonces se podria modificar esta funcion para que ademas de devolver si es o no un malware, devuelva que es

poco confiable segun un rango que se le puede definir. En caso de **c++** que se utiliza la funcion interna de **amis.h** habria que dejar de usar esta funcion y usar una funcion equivalente redefinida como se hace en la aplicacion de C#.

Funciones de AMSI:

AmsiInitialize Inicializa la API de AMSI.

```
HRESULT AmsiInitialize(
  [in]  LPCWSTR      appName,
  [out] HAMSICONTEXT *amsiContext
);
```

`[in] appName`
Nombre, versión o cadena GUID de la aplicación que llama a la API de AMSI.

`[out] amsiContext`
Identificador de tipo HAMSICONTEXT que se debe pasar a todas las llamadas posteriores a la API de AMSI.

Si esta función se realiza correctamente, devuelve **S_OK (0)**. De lo contrario, devuelve un código de error **HRESULT** . En caso de C# se analiza si devuelve 0, que seria el equivalente a S_OK en el enumerable HRESULT de **c++.**

AmsiOpenSession Abre una sesión en la que se pueden correlacionar varias solicitudes de examen. Recibe como entrada la amsiContext que fue salida de la funcion AmsiInitialize y como salida un enumerable HAMSISESSION.

```
HRESULT AmsiOpenSession(
  [in]  HAMSICONTEXT amsiContext,
  [out] HAMSISESSION *amsiSession
);
```

Esta salida Recibera posteriormente a la funcion AmsiScanBuffer.

AmsiCloseSession Cierra una sesión abierta por AmsiOpenSession.

AmsiScanBuffer Examina un búfer lleno de contenido para el malware.

```
HRESULT AmsiScanBuffer(
  [in]          HAMSICONTEXT amsiContext,
  [in]          PVOID        buffer,
  [in]          ULONG        length,
  [in]          LPCWSTR      contentName,
  [in, optional] HAMSISESSION amsiSession,
  [out]         AMSI_RESULT  *result
);
```

De las anteriores funciones se obtienen los datos de entrada para esta funcion de scanear el buffer:

[in] **HAMSICONTEXT amsiContext**: Recibe la HAMSICONTEXT devuelta en la funcion AmsiInitialize.
[in] **PVOID** **buffer**: Recibe un array de bytes, dado un archivo este se convierte a bytes

y Recibe a esta funcion.
[in] **ULONG** **length**: Recibe el length del array de bytes como entero.
[in] **LPCWSTR** **contentName**: Recibe un string nombre del archivo como identificador, el nombre del contenido que se esta analizando.

[in, optional] HAMSISESSION amsiSession: Recibe la HAMSISESSION que fue previamente calculada.

[out] **AMSI_RESULT *result** : devuelve un enumerable(entero) AMSI_RESULT, el cual es resultado de scanear un buffer utilizando esta API.

Si esta función se realiza correctamente, devuelve **S_OK (0)**. De lo contrario, devuelve un código de error **HRESULT** . En caso de C# se analiza si devuelve 0, que seria el equivalente a S_OK en el enumerable HRESULT de **c++.**

AmsiResultIsMalware Determina si el resultado de un examen indica que se debe bloquear el contenido.

```
void AmsiResultIsMalware(
  [in]  result
);
```

[in] result: Recibe el resultado del scaner previo.

AmsiScanString Examina un string para detectar malware.

```
HRESULT AmsiScanString(
  [in]          HAMSICONTEXT amsiContext,
  [in]          LPCWSTR      string,
  [in]          LPCWSTR      contentName,
  [in, optional] HAMSISESSION amsiSession,
  [out]         AMSI_RESULT  *result
);
```

[in] **LPCWSTR** **string:** Recibe un string para ser analizado y definir su AMSI_RESULT.

Las otras entradas y la salida son analogas a la funcion AmsiScanBuffer.

AmsiUninitialize Elimina la instancia de la API de AMSI que AmsiInitialize abrió originalmente.

Aplicacion de C#

Nuestra aplicacion cuenta con un .cs que se encarga de importar los .dll desde la api de AMSI hasta nuestra aplicacion de C#. Esto se realiza en script **AmsiWrapper.cs**, se definen entonces las funciones de amsi que se utilizaran.

Lo mas imporante en la aplicacion es la clase **MalwareScanner.**
En esta se encuentran las funciones principales que se encargan de analizar un archivo y determinar si presenta o no malware. Las funciones mas importantes para ello son las siguientes:

public MalwareScanner (string applicationName){
Este es el inicializador de la clase MalwareScanner, en el mismo Recibe el nombre que tendra la aplicacion y se inicializa la amsi con este nombre, usando la funcion **AmsiInitialize,** sae guarda el amsiContext para utilizarlo posteriormente.
}

public bool HasVirus (string str, string filename){
Recibe un string, este se usa en la funcion **AmsiScanString**, el resultado de esta funcion Recibe a la funcion **HasMalware** y la misma devuelve si es o no malware.
Tambien se abre una sesion con **AmsiOpenSesion**, la cual Recibe junto con el string a la funcion **AmsiScanString.**
}

public bool HasVirus (Stream stream, string filename){
Recibe un Stream (lo que seria un archivo a nivel de C#), este lo convierte a array de bytes con una funcion auxiliar. Abre una sesion con **AmsiOpenSesion**, se devuelve la funcion:

```
byte[] bytearray  = array de bytes que se obtuvo de la conversion;
IntPtr  sesion = sesion que se abrio.
return Scan(bytearray, filename,sesion)
```

}

private int Scan(byte[] bytearray, string filename, IntPtr session){
length = el length del array de bytes;
result = AmsiScanBuffer (amsiContext, bytearray, length, filename, session, out scanResult);

Si result == 0(**S_OK**), llama la funcion HasMalware con entrada scanResult y devuelve la misma; en otro caso lanza una exepcion de error.

```
return HasMalware(scanResult);
```

}

```
private static bool HasMalware (int scanResult){
```

[Explicada mas en detalle en la pagina 1 del informe]
Recibe un entero y analiza el intervalo en el cual esta para devolver si es un malware o no. Y en casos especiales lanzar exepcion.
}

Un ejemplo de entrada a la aplicacion es el siguiente:

```
MalwareScanner scanner = new MalwareScanner("MyApplications Viruscanner");
string url = "C:\\....\\Downloads\\eicar_com.zip";
Stream stream = File.OpenRead(url);
bool result = scanner.HasVirus(stream, "filename");
Console.WriteLine("Malware = " + result.ToString());
```