

Forum:[Hack Forums](#)

Thread: [c# ported EOF class](#)

DragQueen - April 6, 2011, 2:08 pm

```
//
// Now its support native and managed [both of them]
// i added credits in source :P
//

Code class Eof
{
    // here a sample [ Copy & Uncomment it ]
    // |-->
    // byte[] data = Eof.GetEOF(@"file location");
    // if (data != null)
    // {
    //     // to do
    // }
    // <--|

    // look here for more explain
    // http://stackoverflow.com/questions/855756/difference-between-native-and-managed-code

    // source :: https://hackforums.net/showthread.php?tid=1193297
    // auther :: Zer0Fact0
    public static byte[] GetManagedEof(string path)
    {
        try
        {
            byte[] buffer = null;
            using (FileStream fs = new FileStream(path, FileMode.Open))
            {
                buffer = new byte[4];
                fs.Position = 0x3C;
                fs.Read(buffer, 0, 4);
                fs.Position = BitConverter.ToInt32(buffer, 0) + 0x6;
                buffer = new byte[2];
                fs.Read(buffer, 0, 2);
                fs.Position += 0x100 + ((BitConverter.ToInt16(buffer, 0) - 1) * 0x28);
                buffer = new byte[8];
                fs.Read(buffer, 0, 8);
                fs.Position = BitConverter.ToInt32(buffer, 0) + BitConverter.ToInt32(buffer, 4);
                buffer = new byte[fs.Length - fs.Position];
                fs.Read(buffer, 0, buffer.Length);
            }
            return buffer;
        }
        catch { }
    }
}
```

```

    return null;
}

// source :: https://hackforums.net/showthread.php?tid=717991
// auther :: pr0t0typ3
// ported to c# by DragQueen
public static byte[] GetNativeEof(string path)
{
    try
    {
        byte[] buffer = null;
        byte[] data = File.ReadAllBytes(path);
        long size = CalculateSize(data);

        if (size != data.Length)
        {
            buffer = new byte[data.Length - size - 3]; ;
            Array.Copy(data, size + 2, buffer, 0, buffer.Length);
        }
        if (buffer.Length > 0)
            return buffer;
        }
        catch { }
        return null;
    }
    static long CalculateSize(byte[] data)
    {
        long size = 0L;
        GCHandle GC = GCHandle.Alloc(data, GCHandleType.Pinned);
        int buffer = GC.AddrOfPinnedObject().ToInt32();
        DOS_Header DH = (DOS_Header)Marshal.PtrToStructure(GC.AddrOfPinnedObject(),
        typeof(DOS_Header));
        GC.Free();
        NT_Headers NH = (NT_Headers)Marshal.PtrToStructure(new IntPtr(buffer + DH.Address),
        typeof(NT_Headers));
        long offset = DH.Address + 248;
        int length = NH.File.Sections - 1;

        for (int i = 0; i <= length; i++)
        {
            Section_Header header2 = (Section_Header)Marshal.PtrToStructure(new IntPtr((buffer + offset)
+ (i * 40)), typeof(Section_Header));
            size += header2.Size;
        }
        return (size + 4096);
    }
    [StructLayout(LayoutKind.Sequential)]
    struct Data_Directory
    {
        public uint Address;
        public uint Size;
    }
}

```

```

[StructLayout(LayoutKind.Sequential)]
struct DOS_Header
{
    public ushort Magic;
    public ushort Last;
    public ushort Pages;
    public ushort Relocations;
    public ushort Size;
    public ushort Minimum;
    public ushort Maximum;
    public ushort SS;
    public ushort SP;
    public ushort Checksum;
    public ushort IP;
    public ushort CS;
    public ushort Table;
    public ushort Overlay;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
    public ushort[] ReservedA;
    public ushort ID;
    public ushort Info;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 10)]
    public ushort[] ReservedB;
    public int Address;
}
[StructLayout(LayoutKind.Sequential)]
struct File_Header
{
    public ushort Machine;
    public ushort Sections;
    public uint Stamp;
    public uint Table;
    public uint Symbols;
    public ushort Size;
    public ushort Flags;
}
[StructLayout(LayoutKind.Sequential)]
struct Misc
{
    public uint Address;
    public uint Size;
}
[StructLayout(LayoutKind.Sequential)]
struct NT_Headers
{
    public uint Signature;
    public File_Header File;
    public Optional_Headers Optional;
}
[StructLayout(LayoutKind.Sequential)]
struct Optional_Headers
{
    public ushort Magic;

```

```

public byte Major;
public byte Minor;
public uint SCode;
public uint IData;
public uint UData;
public uint Address;
public uint Code;
public uint Data;
public uint Image;
public uint SectionA;
public uint FileA;
public ushort MajorO;
public ushort MinorO;
public ushort MajorI;
public ushort MinorI;
public ushort MajorS;
public ushort MinorS;
public uint Version;
public uint SImage;
public uint SHeaders;
public uint Checksum;
public ushort Subsystem;
public ushort Flags;
public uint SSReserve;
public uint SSCommit;
public uint SHReserve;
public uint SHCommit;
public uint LFlags;
public uint Count;
[MarshalAs(UnmanagedType.ByValArray, SizeConst = 0x10)]
public Data_Directory[] DataDirectory;
}
[StructLayout(LayoutKind.Sequential)]
struct Section_Header
{
    public byte Name;
    public Misc Misc;
    public uint Address;
    public uint Size;
    public uint Pointer;
    public uint PRelocations;
    public uint PLines;
    public uint NRelocations;
    public uint NLines;
    public uint Flags;
}
}

```

siimon00 - April 6, 2011, 3:11 pm

(Apr 6, 2011 02:08 PM)DragQueen Wrote: [CITE](#) if u want get the eof ... so here ported to c# + handle execption + nice code + nice comment based on vb.code from 'https://hackforums.net/showthread.php?tid=717991'

Code class Eof

```
{
// ported to c# from
// https://hackforums.net/showthread.php?tid=717991

// worked on Native exe file's only [c, c++]
// managed exe file's [.net / java] no supported
// http://stackoverflow.com/questions/855756/difference-between-native-and-managed-code

// here a sample [ Copy & Uncomment it ]
// |-->
// byte[] data = Eof.GetEOF(@"file location");
// if (data != null)
// {
// // to do
// }
// <--|

// Methods
public static byte[] GetEOF(string path)
{
    try
    {
        byte[] buffer = null;
        byte[] data = File.ReadAllBytes(path);
        long size = CalculateSize(data);

        if (size != data.Length)
        {
            buffer = new byte[data.Length - size - 3]; ;
            Array.Copy(data, size + 2, buffer, 0, buffer.Length);
        }
        if (buffer.Length > 0)
            return buffer;
        }
        catch { }
        return null;
    }

    static long CalculateSize(byte[] data)
    {
        long size = 0L;
        GCHandle GC = GCHandle.Alloc(data, GCHandleType.Pinned);
        int buffer = GC.AddrOfPinnedObject().ToInt32();
        DOS_Header DH = (DOS_Header)Marshal.PtrToStructure(GC.AddrOfPinnedObject(),
typeof(DOS_Header));
        GC.Free();
        NT_Headers NH = (NT_Headers)Marshal.PtrToStructure(new IntPtr(buffer + DH.Address),
typeof(NT_Headers));
        long offset = DH.Address + 248;
        int length = NH.File.Sections - 1;

        for (int i = 0; i <= length; i++)
```

```

    {
        Section_Header header2 = (Section_Header)Marshal.PtrToStructure(new IntPtr((buffer + offset)
+ (i * 40)), typeof(Section_Header));
        size += header2.Size;
    }
    return (size + 4096);
}

```

// Nested Types

[StructLayout(LayoutKind.Sequential)]

struct Data_Directory

```

{
    public uint Address;
    public uint Size;
}

```

[StructLayout(LayoutKind.Sequential)]

struct DOS_Header

```

{
    public ushort Magic;
    public ushort Last;
    public ushort Pages;
    public ushort Relocations;
    public ushort Size;
    public ushort Minimum;
    public ushort Maximum;
    public ushort SS;
    public ushort SP;
    public ushort Checksum;
    public ushort IP;
    public ushort CS;
    public ushort Table;
    public ushort Overlay;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
    public ushort[] ReservedA;
    public ushort ID;
    public ushort Info;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 10)]
    public ushort[] ReservedB;
    public int Address;
}

```

[StructLayout(LayoutKind.Sequential)]

struct File_Header

```

{
    public ushort Machine;
    public ushort Sections;
    public uint Stamp;
    public uint Table;
    public uint Symbols;
    public ushort Size;
    public ushort Flags;
}

```

```
[StructLayout(LayoutKind.Sequential)]
struct Misc
{
    public uint Address;
    public uint Size;
}
```

```
[StructLayout(LayoutKind.Sequential)]
struct NT_Headers
{
    public uint Signature;
    public File_Header File;
    public Optional_Headers Optional;
}
```

```
[StructLayout(LayoutKind.Sequential)]
struct Optional_Headers
{
    public ushort Magic;
    public byte Major;
    public byte Minor;
    public uint SCode;
    public uint IData;
    public uint UData;
    public uint Address;
    public uint Code;
    public uint Data;
    public uint Image;
    public uint SectionA;
    public uint FileA;
    public ushort MajorO;
    public ushort MinorO;
    public ushort MajorI;
    public ushort MinorI;
    public ushort MajorS;
    public ushort MinorS;
    public uint Version;
    public uint SImage;
    public uint SHeaders;
    public uint Checksum;
    public ushort Subsystem;
    public ushort Flags;
    public uint SSReserve;
    public uint SSCommit;
    public uint SHReserve;
    public uint SHCommit;
    public uint LFlags;
    public uint Count;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 0x10)]
    public Data_Directory[] DataDirectory;
}
```

```
[StructLayout(LayoutKind.Sequential)]
struct Section_Header
{
    public byte Name;
    public Misc Misc;
    public uint Address;
    public uint Size;
    public uint Pointer;
    public uint PRelocations;
    public uint PLines;
    public uint NRelocations;
    public uint NLines;
    public uint Flags;
}
```

FINALLY!

Someone who's actually posting something usefull instead of crypter/stubs, booters, youtube viewers and all that shite!

thank you very much dude, this is sure nicely done of you and will help a lot of people when coding..

I appreciate one actually writing in C# among all these VB users (not to offend anyone, theres just TOO MANY in here)

You have my gratitude m8!

-Siimon ?

DragQueen - April 6, 2011, 8:35 pm

but i still think it will not work

why ? bcz its .net file

and its diffrenct than c / c+ file

So i dont think it will work at all

Ethereal - April 6, 2011, 11:19 pm

(Apr 6, 2011 08:35 PM)DragQueen Wrote: [\[CITE\]](#) but i still think it will not work

why ? bcz its .net file

and its diffrenct than c / c+ file

So i dont think it will work at all

If you know what you are doing, you can crypt C++ with .NET.

Not sure why you would want to, though.

DragQueen - April 7, 2011, 6:50 am

(Apr 6, 2011 11:19 PM)Ethereal Wrote: [\[CITE\]](#) If you know what you are doing, you can crypt C++ with .NET.

Not sure why you would want to, though.

i mean that if ur file c/c+ it can detect this new extra bytes

but if ur file is .net so even u add him some extra data

its no native file ... so how execly the crypted file should fild this extra data ?

iamhere - April 7, 2011, 7:01 am

Code public string ReadEofData(string filePath)

```

{
    var readFile = File.ReadAllText(filePath);
    var IPos = (Strings.StrReverse(readFile)).IndexOf(GetNullBytes(30)) + 1;
    var eofBuf = Strings.StrReverse(readFile).Substring(0, Convert.ToInt32(IPos - 1));
    var returnValue = Strings.StrReverse(eofBuf);
    switch (returnValue)
    {
    case "":
        break;
    }
    return returnValue;
}

public string GetNullBytes(object INum)
{
    var buf = "";

    short i;
    for (i = 1; i <= Convert.ToInt16(INum); i++)

    buf = buf + ControlChars.NullChar;

    var returnValue = buf;
    return returnValue;
}

```

DragQueen - April 7, 2011, 7:15 am

(Apr 7, 2011 07:01 AM)iamhere Wrote: [CITE](#) Code public string ReadEofData(string filePath)

```

{
    var readFile = File.ReadAllText(filePath);
    var IPos = (Strings.StrReverse(readFile)).IndexOf(GetNullBytes(30)) + 1;
    var eofBuf = Strings.StrReverse(readFile).Substring(0, Convert.ToInt32(IPos - 1));
    var returnValue = Strings.StrReverse(eofBuf);
    switch (returnValue)
    {
    case "":
        break;
    }
    return returnValue;
}

public string GetNullBytes(object INum)
{
    var buf = "";

    short i;
    for (i = 1; i <= Convert.ToInt16(INum); i++)

    buf = buf + ControlChars.NullChar;

    var returnValue = buf;
    return returnValue;
}

```

ur function return something else than my function
with my finction i get ... microsoft ... --> end
with ur i get junk !

BaussHacker - April 7, 2011, 7:21 am

(Apr 7, 2011 06:50 AM)DragQueen Wrote: [\[CITE\]](#) i mean that if ur file c/c+ it can detect this new extra bytes

but if ur file is .net so even u add him some extra data

its no native file ... so how exacly the crypted file should fild this extra data ?

You english needs some extra data.

DragQueen - April 7, 2011, 7:24 am

Mine sucks dont use it --> use this e genius <https://hackforums.net/showthread.php?tid=1193297>

,