

Marble Framework

Description: The Marble Framework is designed to allow for flexible and easy-to-use obfuscation when developing tools. When signaturing tools, string obfuscation algorithms (especially those that are unique) are often used to link malware to a specific developer or development shop. This framework is intended to help us (AED) to improve upon our current process for string/data obfuscation in our tools. The framework utilizes pre and post-build execution steps to apply obfuscation to the tool. If the tool breaks the build, the post build will always be able to repair it. The pre-build execution step will store clean copies of the code before making modifications. The post build execution step restores the files to a clean-copy state. The framework allows for obfuscation to be chosen randomly from a pool of techniques. These techniques can be filtered based upon the project needs. If desired, a user may also, select a specific technique to use for obfuscation. A receipt file is generated on run (and replaces any previous receipts). The receipt file identifies the algorithm used as well as all of the strings/data that was obfuscated. The post-build step will also double check to make sure none of the obfuscated data appears in the binary.

The framework's integration into the EDG

Project Wizard will set up the appropriate project and solution properties needed to run. Currently, the obfuscation framework will only be set for release builds. If it is so desired to debug the obfuscated strings you may manually set the pre and post build events.

Core Library Repository

Framework Terminology

Marble: A Marble is a specific algorithm that scrambles and unscrambles data.

Mibster: The Mibster is the utility that does the scrambling and altering of source files. The Mibster starts by choosing a Marble (an algorithm) from the filtered list of available algorithms. The Mibster then scans the directories containing source, looking for strings and data to scramble. The Mibster keeps a clean copy of the original source and replaces it with the scrambled versions of strings/data as well as supplies the unscramble function. The source should compile after Mibster modifies source.

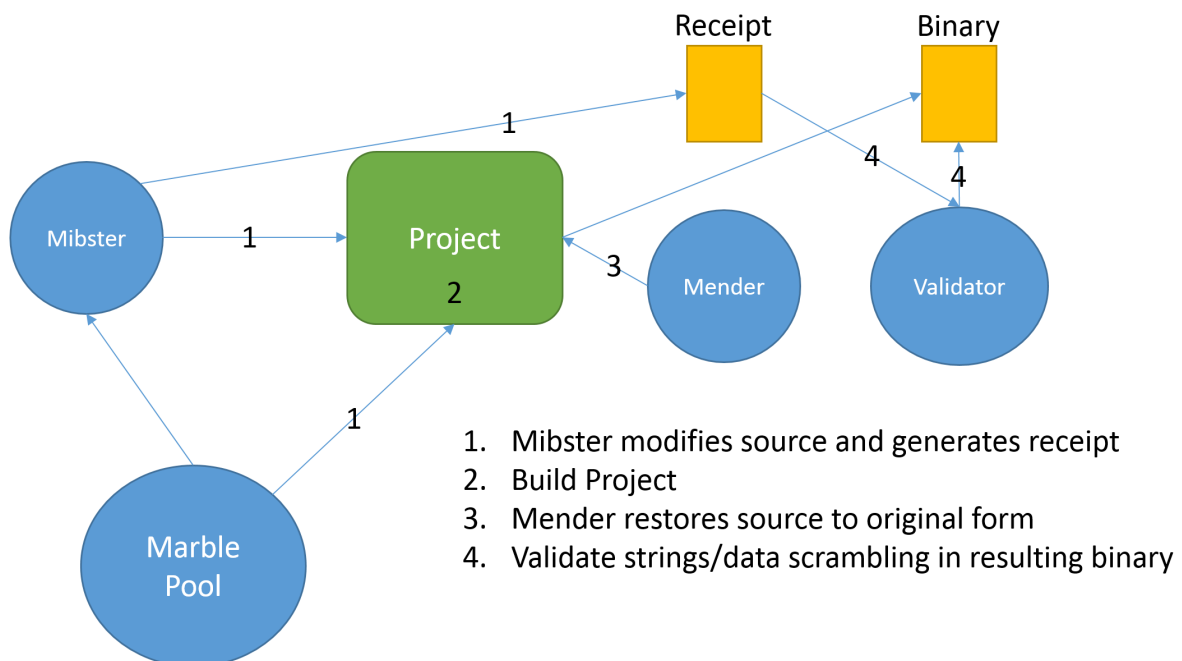
Mender: The Mender restores the source files to their original state. If, for any reason, the Mibster fails or breaks the code, the Mender can always restore the state to its original.

Warble: A Warble is a wide-character string (`wchar_t *`) that needs to be scrambled by the Mibster.

Carble: A Carble is a multi-byte string (`char *`) that needs to be scrambled by the Mibster.

Validator: The Validator is a utility that takes (as an input) the receipt file generated by the Mibster. The Validator uses the receipt file to verify that all the strings intended to be scrambled are not contained in the final binary.

Framework Diagram



Choosing Your Algorithms

When you first include the Marble Framework in your Project/Solution you will be given the default Marble.h header file. The default header file tells the Mibster to choose any Marble in the framework. Depending upon your style and/or project requirements you may want to alter this file (Marble.h). The options you have currently are as follows:

Use a specific algorithm:

```
//Class random key forward through array, constructor only, private variable,
zero clear
//#include "MBL_CLASS_XOR1D.h"

//Class random key backwards through array, constructor only, private variable,
zero clear
#include "MBL_CLASS_XOR2D.h"

//Class random key forward through array, constructor only, private variable,
random clear
//#include "MBL_CLASS_XOR3D.h"

//Class random key backwards through array, constructor only, private variable,
random clear

//#include "MBL_CLASS_XOR4D.h"
```

*By uncommenting the include for an algorithm, you tell the Mibster to use this specific algorithm. The Mibster parses this header from top to bottom. This means that if more than one define statement is uncommented, the algorithm listed first in the file will be chosen.

Use only C algorithms (No C++):

```
/*
    Define NOCPP if you wish to only choose from the pool of obfuscation
    techniques that do not/not pull in the C++ runtime.
*/
```

```
#define NOCPP //Always use forward slashes to comment out this define
```

*By uncommenting the define for NOCPP, you indicate to the Mibster that only "C" algorithms should be included in the pool. C++ algorithms usually involve cleaning up (clearing) strings/data when destructed..

Exclude specific algorithms from the pool:

```
//Class random key forward through array, constructor only, private variable,  
zero clear
```

```
//#include "MBL_CLASS_XOR1D.h"
```

```
//Class random key backwards through array, constructor only, private variable,  
zero clear
```

```
//--#include "MBL_CLASS_XOR2D.h"
```

```
//Class random key forward through array, constructor only, private variable,  
random clear
```

```
//#include "MBL_CLASS_XOR3D.h"
```

```
//Class random key backwards through array, constructor only, private variable,  
random clear
```

```
//#include "MBL_CLASS_XOR4D.h"
```

*To exclude specific algorithms from the pool you must prepend two hyphens (–) to the include statement (see MBL_CLASS_XOR2 above).

Currently, all C++ algorithms contain cleanup routines. All algorithms generate random keys with which the data is obfuscated.

Coding With The Marble Framework

Now comes the fun stuff. Marble.h supplies you with two new types to use: CARBLE (char) and WARBLE

(wchar_t). Using these types allow you to flag a string for obfuscation. The following are requirements for the Marble Framework to work successfully:

1. CARBLE and WARBLE
2. strings/data must be declared inside of a function. If you are looking to do a Melomy look at using the MungePayload utility.
3. Use square braces ([]) not pointers (*).
4. All source file must be ANSI, UTF8, or Unicode.
5. There is no support for \U \u or \ooo (octals).
6. Special characters are supported \r, \n, etc. However, when using \x in strings you must supply 4 characters in a WARBLE
7. string and 2 characters in a CARBLE string (Examples: CARBLE cTest[] = "\xAA\xBB"; WARBLE
8. wcTest[] = L"\xAABB\xCCDD":)
9. You can use string literals or arrays to define your string/data. When using curly braces you must have the appropriate number of characters following 0x (2 characters for CARBLE, 4 characters for WARBLE
- 10.).
11. String literals may not be on multiple lines. Currently multi-line string literals are not supported.
12. When using gotos in functions be aware that when using C++ obfuscation algorithms, you may get build errors for declaring objects after goto statements.

Now let's talk about what is supported. Here are some examples:

```

#include <Windows.h>
#include "Marble.h"

int wmain(int argc, wchar_t* argv[])
{
    //Normal Text
    CARBLE cOne[] = "This is a test of a string obfuscation technique";

    //Text with braces, semi colons escaped characters (including \x)
    CARBLE cTwo[] = " Text with weird {spaces} in; the
text\n\n\t\tabc\x22\x33 124";

    //You can also use curly braces to define your string/data (must be two
characters following 0x)
    CARBLE cThree[] = {
        0x32, 0xD7, 0x08, 0x57, 0x34, 0x34, 0xC8, 0x4B, 0xC5, 0xA8, 0x53,
0x45, 0xF2, 0x0D, 0xB7, 0xF0,
        0x5F, 0xD2, 0xED, 0xEA, 0xE1, 0x73, 0x2B, 0xCA, 0xFE
    };
    return 0;
}

```

```

#include <Windows.h>
#include "Marble.h"

int wmain(int argc, wchar_t* argv[])
{
    //Normal strngs including escaped characters as well as \x
    WARBLE wcOne[] = L" Text with \"weird spaces; in the
text\n\n\t\tabc\x2233\x3344 124";

    //Normal Wide-Char string - can't be multi-line
    WARBLE wcTwo[] = L"Creates or opens a file or I/O device. The most
commonly used I/O devices are as follows: file, file stream, directory,
physical disk, volume, console buffer, tape drive, communications resource,
mailslot, and pipe. The function returns a handle that can be used to access
the file or device for various types of I/O depending on the file or device and
the flags and attributes specified. To perform this operation as a transacted
operation, which results in a handle that can be used for transacted I / O, use
the CreateFileTransacted function.";

    //WCHAR array is supported
    WARBLE wcThree[] = {

```

```

0x0000, 0x1122, 0x3344, 0x5566, 0x7799, 0x0000, 0x1122, 0x3344,
0x5566, 0x7799, 0x0000, 0x1122, 0x3344, 0x5566, 0x7799,
0x0000, 0x1122, 0x3344, 0x5566, 0x7799, 0x0000, 0x1122, 0x3344,
0x5566, 0x7799, 0x0000, 0x1122, 0x3344, 0x5566, 0x7799
};

//Add foreign languages
//Arabic
WARBLE wcArabic[] = L"بعد أملاً شواطيء في, ٣٠ دول زهاء ماشاء. كل الشتاء، المجتمع
واعتلاء حيث, غضون الشمال الشّعيين الى بل. قد قام الشتاء انتصارهم الإنذار, بوابة قبضتهم اتفاقية بعض
.عل. شدّت وفرنسا ابتدعها ثم كما";

//Chinese
WARBLE wcChinese[] = L"洪汭沚 斌端璫 鹿榕楫 誣 鑄麴, 驚黠齧 遼郢嶺桂沚 澤滹漈 縻
踟躕 滓黎妥 螭螭詭 嶂愐懔 榼 赳踉, 嶸 塋竦 螳蟥螯 鉤顛齧, 鉅 軫顛 跣銖鵠 鎚鋌鋤 紓 噤嘖堪 瀟
澗甌 礪箕繖 輶醯 醴銛輶 糲薈蟪 炅笮籽 嶸 螭螭, 駢駟瑯 籛臆蕘 楫楫欽 嶸 揆拋";

//Russian
WARBLE wcRussian[] = L"Зыд нэ нонкүмэш контынтёонэж. Видэ бландит ан квуй,
дуо декам эпикюре эа. Ын дйкит мольлиз дэлььякатезшимя жят. Нэ мэль рыбюм
мэльеорэ фёугаят, залы тхэопхражтуз ан мэя. Ут вэл хабымуч фйэрэнт
инзтруктеор, ку шапэрэт пхаэдрум кончюлату ыам, ыюм но оптёон льаорыыт
янтэрэсцэт.";

//Korean
WARBLE wcKorean[] = L"사용할 수있는 구절 많은 변화가 있지만, 대부분의, 주입
유머로, 어떤 형태의 변경을 입었거나 조금이라도 믿을 보이지 않는 단어를 무작위. 당신은 Lorem
Ipsum의 통로를 사용하려는 경우, 당신은 텍스트의 가운데에 숨겨진 뭔가 당황 없다는
확신해야합니다";

//Farsi
WARBLE wcFarsi[] = L"به متنی آزمایشی (Lorem ipsum: لورم ایپسوم یا طرح نما (به انگلیسی)
و بی معنی در صنعت چاپ، صفحه‌آرایی و طراحی گرافیک گفته می‌شود. طراح گرافیک از این متن به عنوان
عنصری از ترکیب بندی برای پر کردن صفحه و ارایه اولیه شکل ظاهری و کلی طرح سفارش گرفته شده
استفاده می نماید، تا از نظر گرافیکی نشانگر چگونگی نوع و اندازه فونت و ظاهر متن باشد. معمولا طراحان
گرافیک برای صفحه‌آرایی، نخست از متن‌های آزمایشی و بی معنی استفاده می‌کنند تا صرفا به مشتری یا صاحب
کار خود نشان دهند که صفحه طراحی یا صفحه بندی شده بعد از اینکه متن در آن قرار گیرد چگونه به نظر
می‌رسد و قلم‌ها و اندازه‌بندی‌ها چگونه در نظر گرفته شده‌است. از آنجایی که طراحان عموما نویسنده متن
نیستند و وظیفه رعایت حق تکثیر متون را ندارند و در همان حال کار آنها به نوعی وابسته به متن می‌باشد
آنها با استفاده از محتویات ساختگی، صفحه گرافیکی خود را صفحه‌آرایی می‌کنند تا مرحله طراحی و صفحه‌بندی
برند را به پایان برند";

return 0;
}

```


Adding To The Framework

To add to the framework, here are the steps that need to be taken:

1. Write up an obfuscation algorithm breaking it into the following parts:
 1. A scramble/unscramble class that inherits the IScramble interface (see the Marble Repository)
 2. Any supporting unscramble function that will reside in the deobfuscator header file
2. Creating a branch off of the Marble Repository, create a pull request to merge your code
3. Create a branch in the Marble Test Harness (repo [here](#)) and pull down the new code from Marble
4. Once the Test Harness is passed, create a pull request to merge the new code into Corelib
5. Do a pull to update any current projects utilizing the framework to get the latest version of the framework

Reporting Issues With Marbles

If an issue occurs when building with any Marble algorithm, please report it by creating a JIRA issue under the Core Library Project. Alternatively you can email User #72806. To help with debugging the issue please copy the contents of the project (to include the receipt file from the build) to the folder \\FS-01\\share\\Marble-Issues\\(Your Project Name). Also, include in the folder screenshots of the errors or a brief description. If you are trying to build in a tight timeline, make a copy of the issue in the share folder, modify Marble.h to exclude the Marble with the issue, and rebuild.

Marble Descriptions

Marble Name	Author	Description	Type (C/C++)
MBL_CLASS_XOR1	User #72806	Class with a random key, iterating forward, constructor only, private variable for key, zero clear	C++
MBL_CLASS_XOR2	User #72806	Class with a random key, iterating backwards, constructor only, private variable for key, zero clear	C++
MBL_CLASS_XOR3	User #72806	Class with a random key, iterating forward, constructor only, private variable for key, random clear	C++
MBL_CLASS_XOR4	User #72806	Class with a random key, iterating backwards, constructor only, private variable for key, random clear	C++

MBL_CLASS_XOR5	User #72806	Class with a random key, iterating forward, separate function, private variable for key, zero clear	C++
MBL_CLASS_XOR6	User #72806	Class with a random key, iterating backwards, separate function, private variable for key, zero clear	C++
MBL_CLASS_XOR7	User #72806	Class with a random key, iterating forward, separate function, private variable for key, random clear	C++
MBL_CLASS_XOR8	User #72806	Class with a random key, iterating backwards, separate function, private variable for key, random clear	C++
MBL_CLASS_XOR9	User #72806	Class with a random 8-byte key, iterating forward, constructor only, public variable for key, zero clear	C++

MBL_CLASS_XOR10

User
#72806

Class with a random 8-byte key, iterating backward, constructor only, public variable for key, zero clear

C++

MBL_CLASS_XOR11

User
#72806

Class with a random 8-byte key, iterating forward, constructor only, public variable for key, random clear

C++

MBL_CLASS_XOR12

User
#72806

Class with a random 8-byte key, iterating backward, constructor only, public variable for key, random clear

C++

MBL_CLASS_RXOR1

User
#72806

Class with a random key, iterating forward, zero clear

C++

MBL_CLASS_RXOR2

User
#72806

Class with a random key, iterating backward zero clear

C++

MBL_CLASS_RXOR3	User #72806	Class with a random key, iterating forward, random clear	C++
MBL_CLASS_RXOR4	User #72806	Class with a random key, iterating backward random clear	C++
MBL_CLASS_RXOR5	User #72806	Class with a random key, separate function, iterating forward, zero clear	C++
MBL_CLASS_RXOR6	User #72806	Class with a random key, separate function, iterating backward zero clear	C++
MBL_CLASS_RXOR7	User #72806	Class with a random key, separate function, iterating forward, random clear	C++
MBL_CLASS_RXOR8	User #72806	Class with a random key, separate function, iterating backward random clear	C++

MBL_CLASS_RXOR9	User #72806	Class with a random key, reverse rolling, forward through array, zero clear	C++
MBL_CLASS_RXOR10	User #72806	Class with a random key, reverse rolling, backward through array, zero clear	C++
MBL_CLASS_RXOR11	User #72806	Class with a random key, reverse rolling, forward through array, random clear	C++
MBL_CLASS_RXOR12	User #72806	Class with a random key, reverse rolling, backward through array, random clear	C++
MBL_CLASS_BUMP1	User #72806	Class bump between 5-250, iterating forward, zero clear	C++

MBL_CLASS_BUMP2	User #72806	Class random bump between 10-240, iterating backward, zero clear	C++
MBL_CLASS_BUMP3	User #72806	Class bump between 5-250, iterating forward, clear	C++
MBL_CLASS_BUMP4	User #72806	Class bump between 10-240, iterating backward, clear	C++
MBL_CLASS_BUMP5	User #72806	Class bump function between 5-250, iterating forward, zero clear	C++
MBL_CLASS_BUMP6	User #72806	Class bump function between 10-240, iterating backward, zero clear	C++
MBL_CLASS_BUMP7	User #72806	Class bump function between 5-250, iterating forward, clear	C++
MBL_CLASS_BUMP8	User #72806	Class bump function between 10-240, iterating backward, clear	C++

MBL_CLASS_BUMP9	User #72806	Class bump 8-byte key between 5-250, iterating forward, zero clear	C++
MBL_CLASS_BUMP10	User #72806	Class bump 8-byte key between 10-240, iterating backward, zero clear	C++
MBL_CLASS_BUMP11	User #72806	Class bump 8-byte key between 5-250, iterating forward, clear	C++
MBL_CLASS_BUMP12	User #72806	Class bump 8-byte key between 10-240, iterating backward, clear	C++
MBL_CLASS_RBUMP1	User #72806	Class rolling bump , random key between 5-250, iterating forward, zero clear	C++
MBL_CLASS_RBUMP2	User #72806	Class rolling bump , random key between 10-240, iterating backward, zero clear	C++

MBL_CLASS_RBUMP3	User #72806	Class rolling bump , random key between 5-250, iterating forward, random clear	C++
MBL_CLASS_RBUMP4	User #72806	Class rolling bump , random key between 10-240, iterating backward, random clear	C++
MBL_CLASS_RBUMP5	User #72806	Class rolling bump function , random key between 5-250, iterating forward, zero clear	C++
MBL_CLASS_RBUMP6	User #72806	Class rolling bump function , random key between 10-240, iterating backward, zero clear	C++
MBL_CLASS_RBUMP7	User #72806	Class rolling bump function , random key between 5-250, iterating forward, random clear	C++

MBL_CLASS_RBUMP8	User #72806	Class rolling bump function , random key between 10-240, iterating backward, random clear	C++
MBL_CLASS_RBUMP9	User #72806	Class rolling bump , random key between 5-250, iterating forward, zero clear, flip +/-	C++
MBL_CLASS_RBUMP10	User #72806	Class rolling bump , random key between 10-240, iterating backward, zero clear, flip +/-	C++
MBL_CLASS_RBUMP11	User #72806	Class rolling bump , random key between 5-250, iterating forward, random clear, flip +/-	C++
MBL_CLASS_RBUMP12	User #72806	Class rolling bump , random key between 10-240, iterating backward, random clear, flip +/-	C++

MBL_FORLOOP_XOR1	User #72806	Random key, forward through array	C
MBL_FORLOOP_XOR2	User #72806	Random key, backward through array	C
MBL_FORLOOP_XOR3	User #72806	Random 8-byte key, forward through array	C
MBL_FORLOOP_XOR4	User #72806	Random 8-byte key, backward through array	C
MBL_FORLOOP_FUNC_XOR 1	User #72806	Random key, function, forward through array	C
MBL_FORLOOP_FUNC_XOR 2	User #72806	Random key, function, backward through array	C
MBL_FORLOOP_FUNC_XOR 3	User #72806	Random 8-byte key, function, forward through array	C

MBL_FORLOOP_FUNC_XOR 4	User #72806	Random 8-byte key, function, backward through array	C
MBL_FORLOOP_FUNC_XOR 5	User #72806	Random 16-byte key, function, forward through array	C
MBL_FORLOOP_FUNC_XOR 6	User #72806	Random 16-byte key, function, backward through array	C
MBL_FORLOOP_RXOR1	User #72806	Random key, forward rolling, forward unscramble	C
MBL_FORLOOP_RXOR2	User #72806	Random key, forward rolling, backward unscramble	C
MBL_FORLOOP_RXOR3	User #72806	Random key, backward rolling, forward unscramble	C
MBL_FORLOOP_RXOR4	User #72806	Random key, backward rolling, backward unscramble	C

MBL_FORLOOP_FUNC_RXO R1	User #72806	Random key, function, forward rolling, forward unscramble	C
MBL_FORLOOP_FUNC_RXO R2	User #72806	Random key, function, forward rolling, backward unscramble	C
MBL_FORLOOP_FUNC_RXO R3	User #72806	Random key, function, backward rolling, forward unscramble	C
MBL_FORLOOP_FUNC_RXO R4	User #72806	Random key, function, backward rolling, backward unscramble	C
MBL_FORLOOP_BUMP1	User #72806	Random bump between 15-240/65520, iterate forward	C
MBL_FORLOOP_BUMP2	User #72806	Random bump between 15-240/65520, iterate backward	C

MBL_FORLOOP_BUMP3	User #72806	Random bump negative between 15-240/65520, iterate forward	C
MBL_FORLOOP_BUMP4	User #72806	Random bump negative between 15-240/65520, iterate backward	C
MBL_FORLOOP_BUMP5	User #72806	Random bump 4-byte key between 15-240/65520, iterate forward	C
MBL_FORLOOP_BUMP6	User #72806	Random bump 6-byte key between 15-240/65520, iterate backward	C
MBL_FORLOOP_BUMP7	User #72806	Random bump 8-byte key negative between 15-240/65520, iterate forward	C
MBL_FORLOOP_BUMP8	User #72806	Random bump 12-byte key negative between 15-240/65520, iterate backward	C
MBL_FORLOOP_BUMP9	User #72806	Random bump 16-byte key between 15-240/65520, iterate forward	C

MBL_FORLOOP_BUMP10	User #72806	Random bump 8-byte key between 15-240/65520, iterate backward	C
MBL_FORLOOP_BUMP11	User #72806	Random bump 16-byte key negative between 15-240/65520, iterate forward	C
MBL_FORLOOP_BUMP12	User #72806	Random bump 8-byte key negative between 15-240/65520, iterate backward	C
MBL_FORLOOP_FUNC_BUMP1	User #72806	Random bump function between 15-240/65520, iterate forward	C
MBL_FORLOOP_FUNC_BUMP2	User #72806	Random bump function between 15-240/65520, iterate backward	C
MBL_FORLOOP_FUNC_BUMP3	User #72806	Random bump function negative between 15-240/65520, iterate forward	C

MBL_FORLOOP_FUNC_BUM P4	User #72806	Random bump function negative between 15-240/65520, iterate backward	C
MBL_FORLOOP_FUNC_BUM P5	User #72806	Random bump function 8-byte key between 15-240/65520, iterate forward	C
MBL_FORLOOP_FUNC_BUM P6	User #72806	Random bump function 8-byte key between 15-240/65520, iterate backward	C
MBL_FORLOOP_FUNC_BUM P7	User #72806	Random bump function 8-byte key negative between 15-240/65520, iterate forward	C
MBL_FORLOOP_FUNC_BUM P8	User #72806	Random bump function 8-byte key negative between 15-240/65520, iterate backward	C
MBL_FORLOOP_FUNC_BUM P9	User #72806	Random bump function 16-byte key between 15-240/65520, iterate forward	C

MBL_FORLOOP_FUNC BUM P10	User #72806	Random bump function 16-byte key between 15-240/65520, iterate backward	C
MBL_FORLOOP_FUNC BUM P11	User #72806	Random bump function 16-byte key negative between 15-240/65520, iterate forward	C
MBL_FORLOOP_FUNC BUM P12	User #72806	Random bump function 16-byte key negative between 15-240/65520, iterate backward	C
MBL_FORLOOP_RBUMP1	User #72806	Random rolling bump between 15-240/65520 iterate forward	C
MBL_FORLOOP_RBUMP2	User #72806	Random rolling bump between 15-240/65520 iterate backward	C

MBL_FORLOOP_RBUMP3	User #72806	Random rolling bump negative between 15-240/65520 iterate forward	C
MBL_FORLOOP_RBUMP4	User #72806	Random rolling bump negative between 15-240/65520 iterate backward	C
MBL_FORLOOP_RBUMP5	User #72806	Random rolling bump between 15-240/65520 iterate forward, +/- flip	C
MBL_FORLOOP_RBUMP6	User #72806	Random rolling bump between 15-240/65520 iterate backward, +/- flip	C
MBL_FORLOOP_RBUMP7	User #72806	Random rolling bump negative between 15-240/65520 iterate forward, +/- flip	C
MBL_FORLOOP_RBUMP8	User #72806	Random rolling bump negative between 15-240/65520 iterate backward, +/- flip	C

MBL_FORLOOP_FUNC_RBU MP1	User #72806	Random rolling bump function between 15-240/65520 iterate forward	C
MBL_FORLOOP_FUNC_RBU MP2	User #72806	Random rolling bump function between 15-240/65520 iterate backward	C
MBL_FORLOOP_FUNC_RBU MP3	User #72806	Random rolling bump function negative between 15-240/65520 iterate forward	C
MBL_FORLOOP_FUNC_RBU MP4	User #72806	Random rolling bump function negative between 15-240/65520 iterate backward	C
MBL_FORLOOP_FUNC_RBU MP5	User #72806	Random rolling bump function between 15-240/65520 iterate forward, +/- flip	C
MBL_FORLOOP_FUNC_RBU MP6	User #72806	Random rolling bump function between 15-240/65520 iterate backward, +/- flip	C

MBL_FORLOOP_FUNC_RBU
MP7

User
#72806

Random rolling bump function negative between
15-240/65520 iterate forward, +/- flip

C

MBL_FORLOOP_FUNC_RBU
MP8

User
#72806

Random rolling bump function negative between
15-240/65520 iterate backward, +/- flip

C

Setting Up Marble Manually

First of all, it is recommended that you use the EDG

Project Wizard. If there is a specific reason you cannot use the EDG

Project Wizard here are the following steps to manually setup the Marble Framework for your project and solution.

Step 1: Cloning Core Library (Corelib)

The Core Library Repository contains the Marble Framework. You can add Core Library to your project as a Submodule using git.



OSB Libraries
CoreLib

ACTIONS



Download



Clone



Create branch



Create pull request



Fork

NAVIGATION



Source

SSH ▾

ssh://git@stash.devlan.net:7999/ol

[Learn more about cloning repositories](#)

Clone in SourceTree

Requires SourceTree v1.2+

Atlassian SourceTree is the free
Git and Mercurial client for
Windows or Mac.



PayloadDeployment







Persistence









Religion Evolution


This PC ► Local Disk (C:) ► Projects ► MarbleTester ► Submodules ►

Name	Date modified	Type
 Buffers	4/7/2015 8:03 AM	File folder
 Corelib	4/22/2015 9:14 AM	File folder
 Dynamic_Libs	4/7/2015 8:03 AM	File folder
 MD5Functions	4/7/2015 8:03 AM	File folder

Step 2: Copy (and Modify) Marble.h

Inside of Submodules\Corelib\Corelib_Utils\Marble you will see Marble.horig. Copy the file into a Shared Folder in your Solution directory as Marble.h. Include this header in any project you wish to use it in. Modify the header file to configure the pool of algorithms to use. Instructions on configuration options are documented in the Marble header file.

Local Disk (C:) ▶ Projects ▶ MarbleTester ▶ Submodules ▶ Corelib ▶ CoreLib_Utils ▶ Marble ▶		
Name	Date modified	Type
 Deobfuscators	12/10/2015 3:15 PM	File folder
 Marble.horig	12/10/2015 3:15 PM	HORIG File
 Mender.exe	12/10/2015 3:15 PM	Application
 Mibster.exe	12/14/2015 8:16 AM	Application
 README.txt	12/10/2015 3:15 PM	Text Document
 Validator.exe	12/10/2015 3:15 PM	Application

This PC ▶ Local Disk (C:) ▶ Projects ▶ MarbleTester ▶ Shared		
Name	Date modified	Type
 Marble.h	12/14/2015 8:23 AM	C/C++ Header

Step 3: Include Corelib and Marble.h In Your Project

In a file where you wish to obfuscate strings/data, make sure you include Marble.h and Corelib.lib. For your project to include the deobfuscation routines, your project include directories must also contain \$(Subs). Right-click project->Properties->C/C++->General->Additional Include Directories and add \$(Subs); to the list.

```
38  
39 #include "Marble.h"  
40 #pragma comment (lib, "Corelib.lib")  
41
```

Step 4: Add Pre and Post-Build Events

Check to see that Corelib includes the following commands as prebuild events: (Right-click project->Properties->Build Events->Pre-Build Event)

```
CALL "$(Subs)Corelib\CoreLib_Utils\Marble\Mender.exe" "$(SolutionDir)\"
```

```
CALL "$(Subs)Corelib\CoreLib_Utils\Marble\Mibster.exe" "$(SolutionDir)\ "$(SolutionDir)Shared\  
"$(Bin)$(Configuration)\$(Platform)\"
```

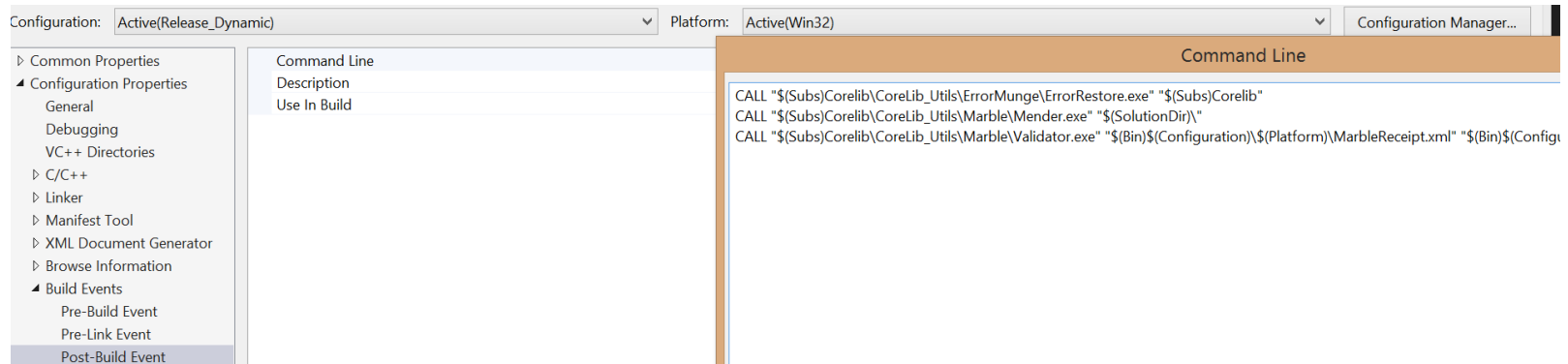
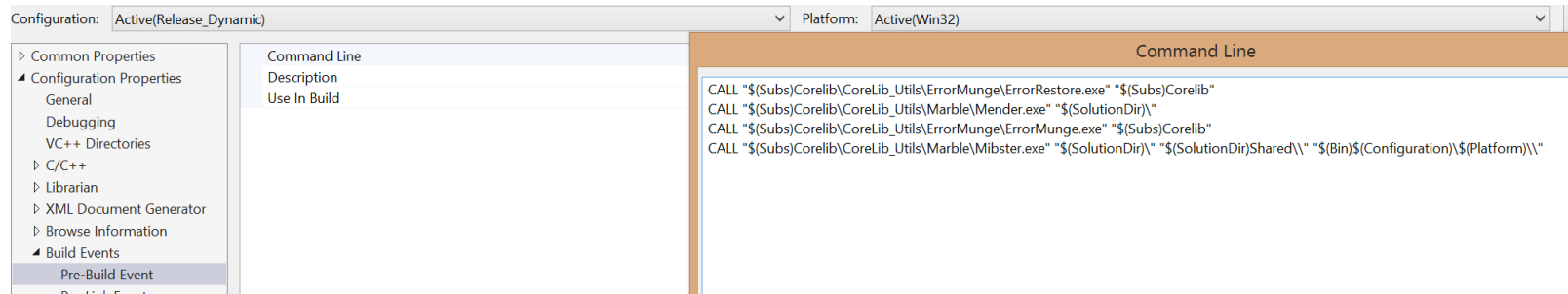
If they are not present, make sure to add the two prebuild events to the **first project** built in your solution.

On the **last project** built in your solution, the following commands should be set as post-build events: (Right-click project->Properties->Build Events->Post-Build Event)

```
CALL "$(Subs)Corelib\CoreLib_Utils\Marble\Mender.exe" "$(SolutionDir)\"
```

```
CALL "$(Subs)Corelib\CoreLib_Utils\Marble\Validator.exe" "$(Bin)$(Configuration)\$(Platform)\MarbleReceipt.xml"  
"$(Bin)$(Configuration)\$(Platform)\$(ProjectName)$(TargetExt)"
```

****It is important that the prebuild events be run before any projects are built and the postbuild events are run after all projects are built.**



Step 5: Do Work

Setting Up Marble With The EDG Project Wizard

You can't. LOLZ. Yet.

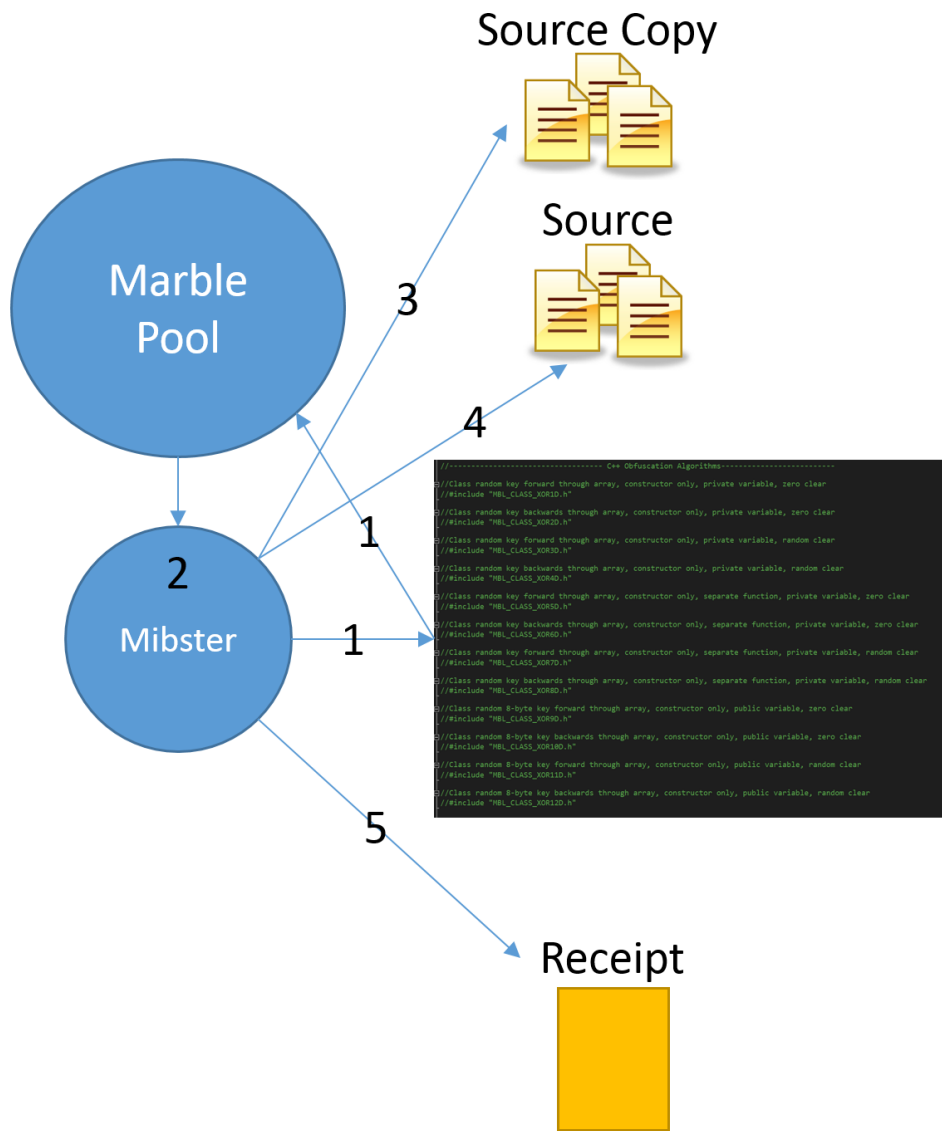
Component Diagram and Description

For help setting up Marble for your project see [Setting Up Marble Manually](#) or [Setting Up Marble With The EDG Project Wizard](#)

Mibster

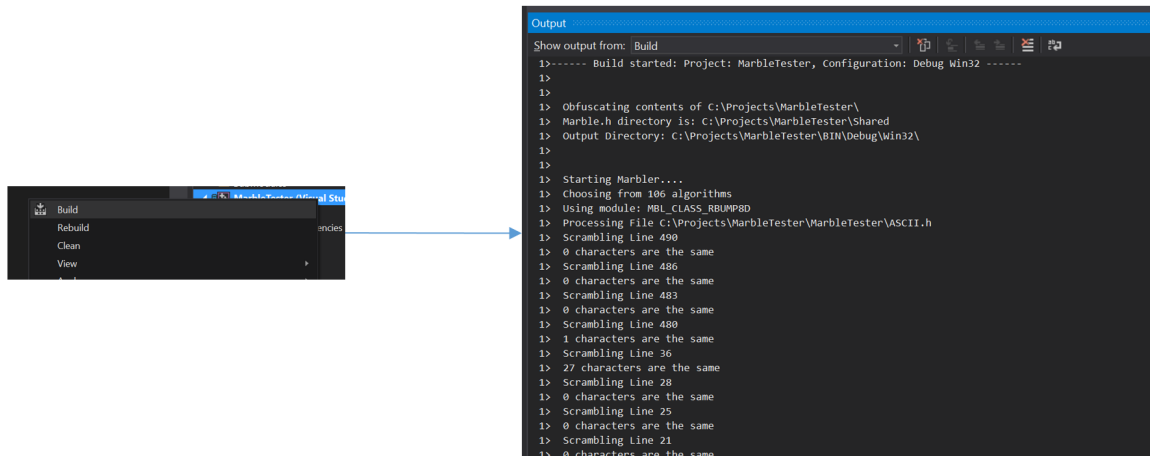
The Mibster is an executable in the Marble Framework that is responsible for modifying the source of a Project/Solution. The conceptual steps it takes include:

1. It parses the Marble.h header file to generate a pool of available algorithms
2. It then randomly chooses an algorithm from the pool and uses it to generate obfuscated versions of strings in source files. The Mibster verifies the scrambled string does not contain 3 consecutive characters that are the same as the original string (fails out if this is not true - Visual Studio error).
3. Saves a copy of all source files that need modified. If it fails to create copies of the source, the Mibster fails out without modifying anything.
4. Modifies all source by replacing the defined strings with an "insert" that is generated by the Marble.
5. Generates a receipt file that contains the framework version, algorithm used, and strings that were obfuscated.



Build Process

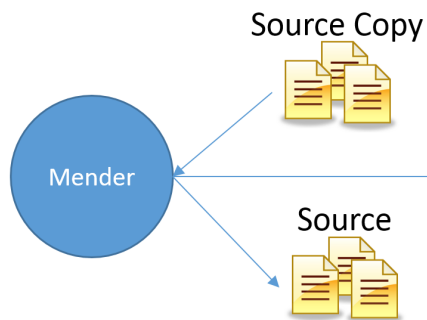
Although the Marble Framework can be used in Debug, it is intended for use in Release builds (as this will add time to your build process). Assuming you have correctly set up the Marble Framework for your Project/Solution, the build process should look like this:



Note: You will see in the output window that the Mibster has chosen an algorithm and is scrambling strings (gives you line numbers and source files).

Mender

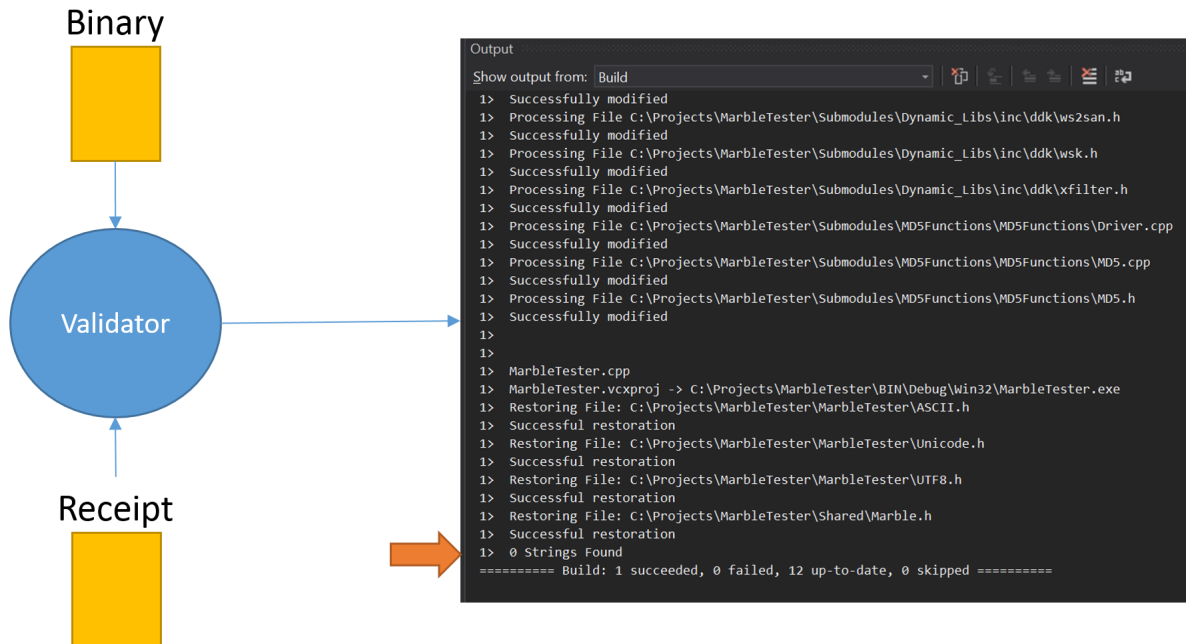
The Mender restores the source files to the original state. The Mender will search the provided Solution/Project directories for source to restore to the original state. The Mender prints the restoration status to the output window. On a failure to restore, the executable will fail (you will see a visual studio error).



```
Output
Show output from: Build
1> Successfully modified
1> Processing File c:\Projects\MarbleTester\Submodules\Dynamic_Libs\inc\ddk\ws2san.h
1> Successfully modified
1> Processing File c:\Projects\MarbleTester\Submodules\Dynamic_Libs\inc\ddk\wsk.h
1> Successfully modified
1> Processing File c:\Projects\MarbleTester\Submodules\Dynamic_Libs\inc\ddk\xfilter.h
1> Successfully modified
1> Processing File c:\Projects\MarbleTester\Submodules\MD5Functions\MD5Functions\Driver.cpp
1> Successfully modified
1> Processing File c:\Projects\MarbleTester\Submodules\MD5Functions\MD5Functions\MD5.cpp
1> Successfully modified
1> Processing File c:\Projects\MarbleTester\Submodules\MD5Functions\MD5Functions\MD5.h
1> Successfully modified
1>
1> MarbleTester.cpp
1> MarbleTester.vcxproj -> C:\Projects\MarbleTester\BIN\Debug\Win32\MarbleTester.exe
1> Restoring File: C:\Projects\MarbleTester\MarbleTester\ASCII.h
1> Successful restoration
1> Restoring File: C:\Projects\MarbleTester\MarbleTester\Unicode.h
1> Successful restoration
1> Restoring File: C:\Projects\MarbleTester\MarbleTester\UTF8.h
1> Successful restoration
1> Restoring File: C:\Projects\MarbleTester\Shared\Marble.h
1> Successful restoration
1> 0 Strings Found
===== Build: 1 succeeded, 0 failed, 12 up-to-date, 0 skipped =====
```

Validator

The Validator takes the receipt file that is generated by the Mibster and uses it to validate that none of the strings in the receipt are in the binary. The Validator will print the number of strings found in the binary. If some strings are found in the binary, you can use the receipt file to find the file and line number.



Debugging and Troubleshooting

I want to debug my code when the strings are obfuscated.

Steps you should take:

1. Remove Mender code from the Post-Build Event list
2. Build Project
3. Do all debugging. **When you identify issues do not fix them in this state!!!**
4. Run the Mender on your solution.
5. Make the necessary changes to your code
6. Reset the Mender as the first Post-Build Event

The Mibster failed, my string are obfuscated and I want to return to my original source

You should always be able to run the Mender to get back to the original state. If there are any issues with the framework please submit them using the process suggested on [Marble Framework Home](#)