OBFUSCATOR

20165110 컴퓨터공학과 김동균

코드 난독기

목차

- Obfuscator란 무엇인가
- Tools & Structure
- 사용해보기
- 장점 & 단점

OBFUSCATION

OBFUSCAOTR 란 무엇인가

우리 신입이.. 뭐라고 좀 했더니 삐져서 이런거 하고 있네.

```
#include <iostream>
#define e using
#define ee namespace
#define eee std
#define eeee ;
#define eeeee int
#define eeeeee main
#define eeeeeee (
#define eeeeeeee )
#define eeeeeeeee while
#define eeeeeeeeee true
#define eeeeeeeeeee {
#define eeeeeeeeeeee }
#define eeeeeeeeeeeee cout
#define eeeeeeeeeeeeee cerr
#define eeeeeeeeeeeeeee <<
#define eeeeeeeeeeeeeeee 'e'
#define eeeeeeeeeeeeeeeee return

e ee eee eeee
eeeee eeeeee eeeeeee eeeeeeee
eeeeeeeeeee
eeeeeeeee eeeeeee eeeeeeeee eeeeeeee
eeeeeeeeeee
eeeeeeeeeeeee eeeeeeeeeeeeee eeeeeeeeeeeeeee eeee
eeeeeeeeeeeeee eeeeeeeeeeeeee eeeeeeeeeeeeeee eeee
eeeeeeeeeee
eeeeeeeeeeeeeeee eeeeeeeeeeeeeee eeee
eeeeeeeeeee
```

?

https://obfuscator.io

obfuscator.io

JavaScript Obfuscator Tool

A free and efficient obfuscator for JavaScript (including partial support of ES2019). Make your code harder to copy and prevent people from stealing your work. This tool is a Web UI to the excellent (and open source) `javascript-obfuscator@3.0.0` created by Timofey Kachalov.

Star 7,983 Watch Sponsor

What is this?

This tool transforms your original JavaScript source code into a new representation that's harder to understand, copy, re-use and modify without authorization. The obfuscated result will have the exact functionality of the original code.

So, it is like UglifyJS, Closure Compiler, etc?

Yes and no. While UglifyJS (and others minifiers) does make the output code harder to understand (compressed and ugly), it can be easily transformed into something readable using a JS Beautifier.

This tool prevents that by using various transformations and "traps", such as **self-defending** and **debug protection**.

How does the obfuscation work?

Through a series of transformations, such as variable / function / arguments renaming, string removal, and others, your source code is transformed into something unreadable, while working exactly as before.

Read more in the FAQ...

Sounds great!

Just paste your code or upload it below and click on "obfuscate".

Also, be sure to read about all the options to understand all the trade-offs between code protection and code size / speed.

Copy & Paste JavaScript Code | Upload JavaScript File | Output

```
1 // Paste your JavaScript code here
2 function hi() {
3   console.log("Hello World!");
4 }
5 hi();
```

Obfuscate

Reset options

Strings Transformations

Identifiers Transformations

Other Transformations

https://picheta.me/obfuscator

picheta.me

OBFUSCATOR

Obfuscate C & C++

Right in your browser!

C++ | Fibonacci

Rename identifiers

```cpp
#include <iostream>

int main()
{
    unsigned int a = 1, b = 1;
    unsigned int target = 46;
    for(unsigned int n = 3; n <= target; ++n)
    {
        unsigned int fib = a + b;
        std::cout << "F("<< n << ") = " << fib << std::endl;
        a = b;
        b = fib;
    }

    return 0;
}
```

```cpp
#include <iostream>
int main(){unsigned int a=(0x0000000000000002 + 0x0000000000000201 +
    0x0000000000000801 - 0x0000000000000A03),b=(0x0000000000000002 +
    0x0000000000000201 + 0x0000000000000801 - 0x0000000000000A03);unsigned int
    target=(0x000000000000005C + 0x000000000000022E + 0x000000000000082E -
    0x0000000000000A8A);for (unsigned int n=(0x0000000000000006 +
    0x0000000000000203 + 0x0000000000000803 - 0x0000000000000A09);(n <= target) &
    !!(n <= target);++n){unsigned int fib=a + b;std::cout<<"\x46""("<<n<<"\x29""
    \075 "<<fib<<std::endl;a = b;b = fib;};return (0x0000000000000000 +
    0x0000000000000200 + 0x0000000000000800 - 0x0000000000000A00);};
```

Obfuscate | Clear

Obfuscation Log

```
Writing /var/obfuscator/code_624a39c24e2794e907566793c1c89b53.obf.cpp
```

https://picheta.me/obfuscator

picheta.me

OBFUSCATOR

Obfuscate C & C++

Right in your browser!

C++ | Fibonacci

Rename identifiers

```
#include <iostream>

int main()
{
    unsigned int a = 1, b = 1;
    unsigned int target = 46;
    for(unsigned int n = 3; n <= target; ++n)
    {
        unsigned int fib = a + b;
        std::cout << "F("<< n << ") = " << fib << std::endl;
        a = b;
        b = fib;
    }

    return 0;
}
```

```
#include <iostream>
int main(){unsigned int o_e448284585366067d4777ca36d402679=(0x0000000000000002 +
    0x0000000000000201 + 0x0000000000000801 - 0x0000000000000A03
    ),o_f026ab76785d027bfd902f94fcb2f700=(0x0000000000000002 + 0x0000000000000201
    + 0x0000000000000801 - 0x0000000000000A03);unsigned int
    o_22aaeaa4f4009220a64cc2bf305a32e8=(0x000000000000005C + 0x000000000000022E +
    0x000000000000082E - 0x0000000000000A8A);for (unsigned int
    o_92e9dfdedbfbf8b2955e60e7a6f2f10a=(0x0000000000000006 + 0x0000000000000203 +
    0x0000000000000803 - 0x0000000000000A09);(o_92e9dfdedbfbf8b2955e60e7a6f2f10a
    <= o_22aaeaa4f4009220a64cc2bf305a32e8) & !!
    (o_92e9dfdedbfbf8b2955e60e7a6f2f10a <= o_22aaeaa4f4009220a64cc2bf305a32e8
    );++o_92e9dfdedbfbf8b2955e60e7a6f2f10a){unsigned int
    o_1b80a6b8e10246e051a691afd49f6917=o_e448284585366067d4777ca36d402679 +
    o_f026ab76785d027bfd902f94fcb2f700;std::cout<<"\x46""("
    <<o_92e9dfdedbfbf8b2955e60e7a6f2f10a<<"\x29"" \075 "
    <<o_1b80a6b8e10246e051a691afd49f6917<<std::endl
    ;o_e448284585366067d4777ca36d402679 = o_f026ab76785d027bfd902f94fcb2f700
    ;o_f026ab76785d027bfd902f94fcb2f700 = o_1b80a6b8e10246e051a691afd49f6917;}
    ;return (0x0000000000000000 + 0x0000000000000200 + 0x0000000000000800 -
    0x0000000000000A00);};
```

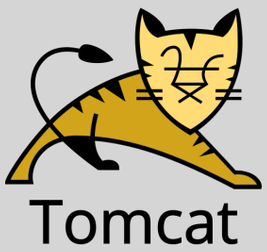
Obfuscate | Clear

Obfuscation Log

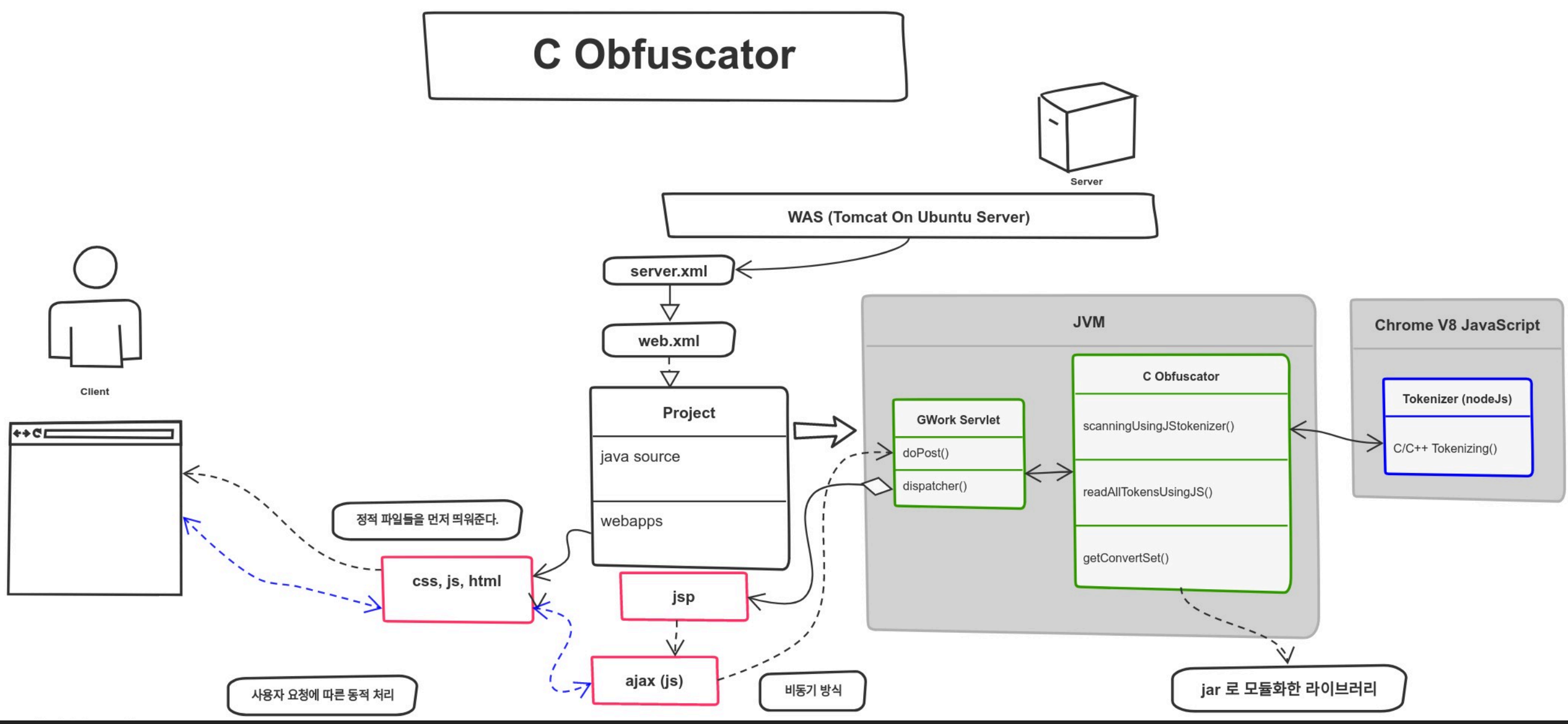
```
Writing /var/obfuscator/code_624a39c24e2794e907566793c1c89b53.obf.cpp
```

TOOLS & STRUCTURE

TOOLS



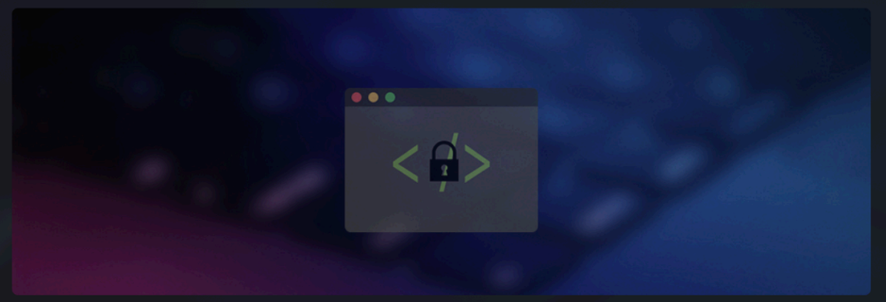
C Obfuscator



View 1 [main page]

View 2 [intro page]

INTRO



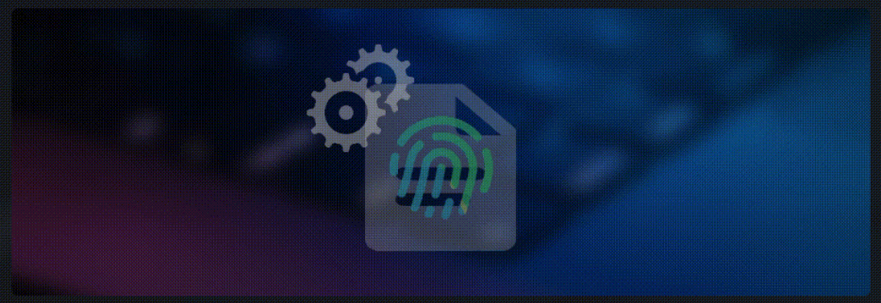
C Obfuscator 는 c 혹은 c++ 소스코드를 난독화 하는 도구입니다. 일반적으로 오픈소스로 배포를 하게 될 경우 일부 소스파일의 경우 난독화가 필요한 경우가 발생할 때 사용할 수 있습니다. 물론 소스코드를 난독화한다고 무조건 소스를 완벽히 보호 할 수는 없겠죠. 하지만 적어도 상대방이 분석 방법을 모를경우 소스코드를 볼 때 분석을 포기할 확률을 높일 수 있는 기대를 할 수 있습니다!

C Obfuscator is a tool that obfuscates c or c++ source code. In general, when distributing as an open source, some source files can be used when obfuscation is required. Of course, obfuscating the source code will not necessarily completely protect the source. However, at least if the other party does not know how to analyze it, you can expect to increase the probability of giving up the analysis when looking at the source code!

사용해보기 >>

View 3 [work page]

WORK



C / C++ 소스코드를 입력하고 제출해보세요! 그러면 하단 박스에 재미있는 소스코드가 나옵니다!

Enter the C / C++ source code and submit it! Then, a fun source code will come out in the box below!

code

SUBMIT CLEAR +

View 4 [about page]

ABOUT

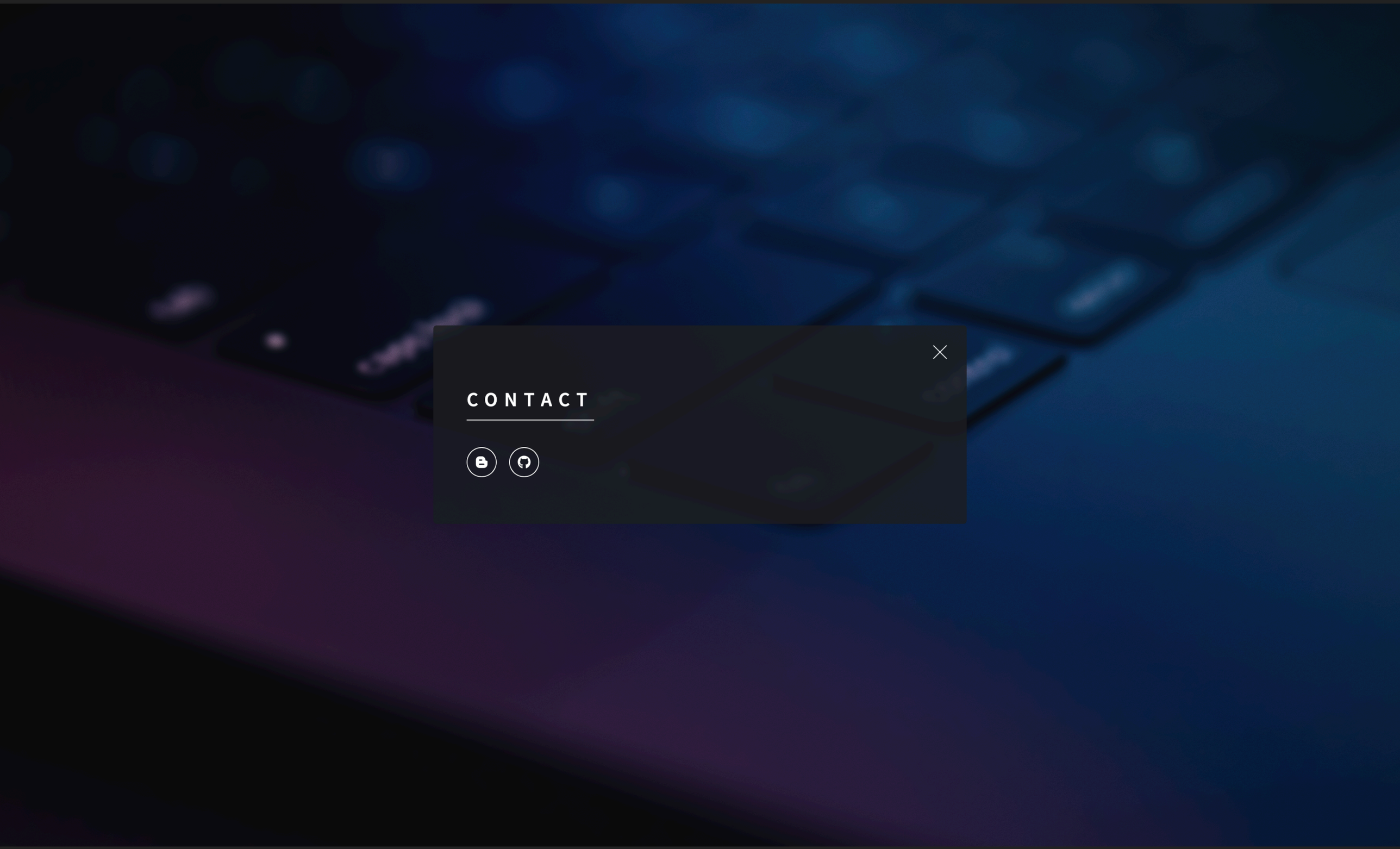
C Obfuscator 를 직접 다뤄볼 수도 있도록 모듈로도 제공합니다! 제공되는 모듈을 직접 다뤄보시고, 기여를 통해 좀 더 높은 퀄리티로 제공 될 수 있도록 함께 도와주시면 감사하겠습니다.

It's also provided as a module so that you can handle it yourself! We would appreciate it if you could handle the modules provided and help us to deliver them in higher quality through contribution.

모듈 둘러보기 >>

Contact >>

View 5 [contact page]



사용해보기

CObfuscator link

test 1 로컬 테스트

test 2 BOJ 별 찍기 - 10

test 3 BOJ 가장 가까운 두 점

장절 & 단절

장점 & 단점

- **장점**
  - 코드 분석을 어렵게 하여 악의적 목적으로 사용하는 것을 방지할 수 있다.
  - 각 코드 별 identity가 존재하기 때문에 복제를 방지할 수 있다.
  - disassemble을 및 decompile을 하더라도 쉽게 해독할 수 있는 것은 아니다.

- **단점**
  - 복잡성에 따라 compile time 및 runtime 성능이 떨어지게 된다.
  - 코드를 변경하는 작업이기 때문에 의존 관계가 복잡해질 수록 정상적 실행이 어렵게 될 가능성이 높다.

THANK YOU.

감사합니다.