StarForce Protection System 2023

StarForce C++ Obfuscator 2.0
Руководство пользователя

Версия 2.0.531
03.03.2023

www.star-force.ru

Оглавление

1 Введение

StarForce C++ Obfuscator предназначен для усложнения анализа программ, написанных на C++, с помощью запутывания их кода. При этом обфускатор не защищает напрямую от модификации кода и не содержит собственных средств обнаружения отладчиков и противодействия им.

StarForce C++ Obfuscator представляет собой программу, которая получает на вход файл с исходным текстом программы на C++ и выдаёт выходной файл, также являющийся исходным текстом программы на C++, который отличается от входного файла тем, что выбранные разработчиком функции (методы классов) преобразованы в нём в функционально эквивалентные с использованием алгоритмов обфускации. Все остальные функции (методы классов) остаются в первоначальном виде.

Если функции достаточно сложные, обфускация очень хорошо защищает от анализа функций в статике и сильно усложняет анализ в динамике. Если функции, подвергающиеся обфускации, содержат много ветвлений, ряд из которых выполняется редко, но при этом они необходимы для работы программы, то даже с использованием статистического анализа и последующей оптимизации получение того же алгоритма будет трудоёмким, и всегда сохраняется высокая вероятность того, что взломщик восстановит алгоритм не полностью.

Синтаксический разбор входного файла производится с помощью базового компилятора C++. Соответствующие модули входят в состав обфускатора, и их не нужно устанавливать отдельно. Базовый компилятор определяет совместимость обфускатора со стандартами языка C++. В версии StarForce C++ Obfuscator 2.0 в качестве базового компилятора используется Clang 14.0.0, который поддерживает стандарты C++98, C++03, C++11, C++14, C++17 и, частично, C++20.

Пример работы обфускатора:

Исходный код	Обфусцированный код

`#include <stdio.h>`
`#include "omni_common.h"`
 	
`// Euclidian algorithm for calculating greatest common divisor (before obfuscation)`
`__attribute__((annotate("StarForce::Obfuscator::Obfuscate(3)")))`
`int gcd( int n, int m )`
`{`
`  if( n < 1 || m < 1 ) return -1;`
`  while( n != m )`
`  {`
`  if( n > m ) n -= m;`
`  else m -= n;`
`  }`
`  return n;`
`}`
 	
`// Tests in triplets { n, m, expected_gcd( n, m ) }`
`int tests[][ 3 ] = {`
`  { 1, 2, 1 },`
`  { 3, 3, 3 },`
`  { 42, 56, 14 },`
`  { 249084, 261183, 111 },`
`};`
 	
`// Perform tests`
`int main( int, char*[] )`
`{`
`  printf( "Performing tests of gcd function:\n" );`
	`#include <stdio.h>`
	`#include "omni_common.h"`
	
	`// Euclidian algorithm for calculating greatest common divisor (before obfuscation)`
	`/// OBFUSCATED BY OMNI OBFUSCATOR V2.0.345, SEED VALUE: 1552483040`
	
	`/// OMNI GENERATED COMMON CODE FOR ALL FUNCTIONS`
	
	`// Obfuscated function`
	`int gcd( int n_2, int m_3 )`
	`{`
	`  int temp_5 = 1;`
	`  int temp_9 = 1;`
	`  int temp_11 = 1;`
	`  bool temp_22 = false;`
	`  unsigned int temp_23 = 0;`
	`  unsigned int temp_24 = 0;`
	`  unsigned int temp_25 = 0;`
	`  unsigned int temp_27 = 0;`
	`  int temp_37 = 0;`
	`  bool temp_38 = false;`
	`  bool temp_39 = false;`
	`  unsigned int temp_40 = 0;`
	`  int temp_41 = 0;`
	`  int temp_42 = 0;`
	`  bool state0_43 = false;`
	`  bool state1_44 = false;`
	`  bool state2_45 = false;`

```
	bool passed = true;
	for( int i = 0; i < sizeof( tests ) /
sizeof( tests[ 0 ] ); i++ )
	{
		int n = tests[ i ][ 0 ];
		int m = tests[ i ][ 1 ];
		int expected_gcd = tests[ i ][ 2 ];
		int calculated_gcd = gcd( n, m );
		printf( "  %d. gcd( %d, %d ) = %
d, ", i + 1, n, m, calculated_gcd );
		if( calculated_gcd ==
expected_gcd )
		{
			printf( "OK.\n" );
		}
		else
		{
			printf( "error.\n" );
			passed = false;
		}
	}
	printf( "Tests %s.\n", passed ? "passed" :
"failed" );
	return passed ? 0 : 1;
}
```

```
	bool state3_46 = false;
	bool state4_47 = false;
	bool state5_48 = false;
	bool state6_49 = false;
	bool state7_50 = false;

	L1:
	L0:
	{
	state0_43 = (bool)1;
	state1_44 = ( bool )( state0_43 == 0 );
	state2_45 = ( bool )( state1_44 == 0 );
	state3_46 = (bool)state2_45;
	goto L76;
	}
	L2:
	{
	temp_27 = ( unsigned int )(temp_38);
	if (state6_49) goto L40; else goto L108;
	}
	L6:
	{
	if (state7_50) goto L78; else goto L110;
	}
	L8:
	{
	temp_27 = ( unsigned int )(temp_41);
	if (state1_44) goto L98; else goto L60;
	}
	L10:
	{
	temp_27 = ( unsigned int )(temp_40);
	state2_45 = ( bool )( state5_48 == 0 );
	if (temp_39) goto L98; else goto L12;
	}
	L12:
	{
	state4_47 = (bool)state7_50;
	temp_38 = temp_37 != temp_41;
	state6_49 = (bool)state0_43;
	if (temp_38) goto L14; else goto L16;
	}
	L14:
	{
	temp_38 = temp_37 > temp_41;
	temp_27 = ( unsigned int )(temp_41);
	if (temp_38) goto L18; else goto L20;
	}
	L16:
	{
	return temp_37;
	}
	L18:
	{
	temp_37 = temp_37 - temp_41;
	goto L96;
	}
	L20:
	{
```

```
            if (state6_49) goto L26; else goto L112;
            }
            L26:
            {
            goto L102;
            }
            L28:
            {
            // The next string is really just an assignment on 32bit
platform
        temp_42 = ( int )( ( size_t )( temp_42 ) + ( ( ( size_t )
        ( temp_42 ) << 31 ) << 1 ) + ( ( ( ( size_t )( temp_42 ) <<
        31 ) << 1 ) >> 15 ) );
            temp_37 = temp_41 + temp_42;
            goto L100;
            }
            L30:
            {
            state0_43 = ( bool )( state3_46 == 0 );
            temp_38 = temp_37 < temp_5;
            temp_40 = (unsigned int)state1_44;
            temp_27 = (unsigned int)state1_44;
            goto L94;
            }
            L32:
            {
            temp_42 = (int)2446449515u;
            // The next string is really just an assignment on 32bit
platform
        temp_42 = ( int )( ( size_t )( temp_42 ) + ( ( ( size_t )
        ( temp_42 ) << 31 ) << 1 ) + ( ( ( ( size_t )( temp_42 ) <<
        31 ) << 1 ) >> 15 ) );
            if (state5_48) goto L82; else goto L104;
            }
            L34:
            {
            if (state4_47) goto L66; else goto L68;
            }
            L36:
            {
            temp_41 = temp_37 - temp_42;
            return temp_41;
            }
            L38:
            {
            temp_39 = ( bool )( state2_45 == 0 );
            if (temp_38) goto L100; else goto L8;
            }
            L40:
            {
            temp_41 =  -temp_11;
            temp_42 = (int)2446449515u;
            temp_27 = ( unsigned int )(temp_40);
            goto L94;
            }
            L42:
            {
            temp_41 =  -temp_11;
            temp_23 = ( unsigned int )(state1_44);
```

```
            temp_24 = (unsigned int)14510u;
            temp_23 = temp_23 * temp_24;
            temp_24 = (unsigned int)2446464025u;
            temp_23 = temp_24 - temp_23;
            temp_42 = ( int )( ( ptrdiff_t )( ( temp_23 ) & 0xFFFFFFFF
) );
            temp_27 = ( unsigned int )(temp_37);
            // The next string is really just an assignment on 32bit
platform
          temp_42 = ( int )( ( size_t )( temp_42 ) + ( ( ( size_t )
          ( temp_42 ) << 31 ) << 1 ) + ( ( ( ( size_t )( temp_42 ) <<
          31 ) << 1 ) >> 15 ) );
            temp_37 = temp_41 + temp_42;
            goto L32;
            }
            L46:
            {
            state1_44 = (bool)state2_45;
            temp_39 = (bool)temp_38;
            if (state2_45) goto L10; else goto L104;
            }
            L48:
            {
            temp_27 = ( unsigned int )(temp_41);
            temp_39 = (bool)temp_38;
            state1_44 = (bool)state6_49;
            goto L50;
            }
            L50:
            {
            temp_27 = ( unsigned int )(temp_37);
            state2_45 = (bool)state0_43;
            if (temp_39) goto L2; else goto L52;
            }
            L52:
            {
            temp_27 = ( unsigned int )(temp_40);
            state6_49 = (bool)state7_50;
            state4_47 = ( bool )( state5_48 == 0 );
            temp_38 = temp_37 != temp_41;
            if (temp_38) goto L54; else goto L104;
            }
            L54:
            {
            temp_38 = temp_37 > temp_41;
            if (temp_38) goto L18; else goto L92;
            }
            L56:
            {
            temp_39 = ( bool )(temp_27);
            return temp_37;
            }
            L58:
            {
            temp_38 = temp_41 < temp_9;
            temp_39 = (bool)temp_38;
            state1_44 = (bool)state2_45;
            goto L96;
            }
```

```
L60:
{
state1_44 = ( bool )( state0_43 == 0 );
temp_38 = temp_41 < temp_9;
temp_39 = (bool)temp_38;
goto L50;
}
L62:
{
temp_41 = temp_41 - temp_37;
if (state5_48) goto L114; else goto L34;
}
L64:
{
temp_27 = ( unsigned int )(temp_41);
temp_41 = temp_41 - temp_37;
if (state4_47) goto L60; else goto L12;
}
L66:
{
temp_40 = (unsigned int)0u;
if (state1_44) goto L52; else goto L74;
}
L68:
{
temp_40 = (unsigned int)0u;
if (state4_47) goto L6; else goto L106;
}
L72:
{
state4_47 = (bool)state2_45;
temp_39 = (bool)0u;
temp_27 = ( unsigned int )(temp_41);
temp_39 = (bool)0u;
temp_27 = ( unsigned int )(temp_37);
if (temp_38) goto L68; else goto L92;
}
L74:
{
temp_39 = (bool)0u;
state4_47 = ( bool )( state0_43 == 0 );
temp_27 = ( unsigned int )(temp_40);
goto L88;
}
L76:
{
state4_47 = ( bool )( state3_46 == 0 );
state5_48 = ( bool )( state4_47 == 0 );
state6_49 = (bool)state5_48;
goto L84;
}
L78:
{
temp_41 = (int)m_3;
temp_37 = (int)n_2;
if (state7_50) goto L26; else goto L90;
}
L80:
{
```

```
if (state7_50) goto L84; else goto L86;
}
L82:
{
state1_44 = (bool)state0_43;
goto L86;
}
L84:
{
state7_50 = ( bool )( state6_49 == 0 );
goto L26;
}
L86:
{
temp_39 = ( bool )(temp_27);
if (state4_47) goto L104; else goto L34;
}
L88:
{
if (state0_43) goto L78; else goto L38;
}
L90:
{
temp_27 = ( unsigned int )(temp_38);
if (state6_49) goto L30; else goto L66;
}
L92:
{
if (state4_47) goto L8; else goto L20;
}
L94:
{
if (state1_44) goto L28; else goto L80;
}
L96:
{
temp_40 = ( unsigned int )(temp_37);
temp_27 = ( unsigned int )(temp_37);
temp_38 = ( bool )(temp_27);
if (state6_49) goto L10; else goto L52;
}
L98:
{
if (state1_44) goto L102; else goto L58;
}
L100:
{
if (state2_45) goto L6; else goto L32;
}
L102:
{
if (state0_43) goto L88; else goto L2;
}
L104:
{
if (state1_44) goto L56; else goto L36;
}
L106:
{
```

```
		temp_27 = ( unsigned int )(temp_41);
		if (state1_44) goto L64; else goto L72;
		}
		L108:
		{
		if (state1_44) goto L42; else goto L48;
		}
		L110:
		{
		temp_27 = ( unsigned int )(temp_39);
		if (state1_44) goto L46; else goto L108;
		}
		L112:
		{
		if (state2_45) goto L62; else goto L106;
		}
		L114:
		{
		temp_27 = ( unsigned int )(temp_39);
		goto L90;
		}
}


// Tests in triplets { n, m, expected_gcd( n, m ) }
int tests[][ 3 ] = {
		{ 1, 2, 1 },
		{ 3, 3, 3 },
		{ 42, 56, 14 },
		{ 249084, 261183, 111 },
};

// Perform tests
int main( int, char*[] )
{
		printf( "Performing tests of gcd function:\n" );
		bool passed = true;
		for( int i = 0; i < sizeof( tests ) / sizeof( tests[ 0 ] ); i++ )
		{
			int n = tests[ i ][ 0 ];
			int m = tests[ i ][ 1 ];
			int expected_gcd = tests[ i ][ 2 ];
			int calculated_gcd = gcd( n, m );
			printf( "  %d. gcd( %d, %d ) = %d, ", i + 1, n, m,
calculated_gcd );
			if( calculated_gcd == expected_gcd )
			{
				printf( "OK.\n" );
			}
			else
			{
				printf( "error.\n" );
				passed = false;
			}
		}
		printf( "Tests %s.\n", passed ? "passed" : "failed" );
		return passed ? 0 : 1;
}
```

Пример работы обфускатора

2 Быстрый старт

Для того чтобы начать использовать обфускатор, выполните следующие шаги:

1. Подготовьте исходный файл программы на C++, который вы можете скомпилировать и затем запустить.
2. Скопируйте дистрибутив StarForce C++ Obfuscator в какую-либо папку на вашем компьютере. Найдите в скопированном дистрибутиве файл Obfuscator.exe. Это единственный исполняемый файл обфускатора, с которым вам придётся работать. Obfuscator.exe является консольным приложением, поэтому дальнейшую работу с ним удобнее проводить из консоли (например, с помощью cmd.exe).
3. Активируйте лицензию на обфускатор:
   a. Запустите файл Obfuscator.exe.
   b. Следуйте инструкциям в появляющихся окнах и активируйте лицензию с помощью серийного номера, полученного от поставщика обфускатора.

      После активации файл Obfuscator.exe будет запускаться без дополнительных запросов на подтверждение лицензии.
4. Подготовьте ваш исходный файл на C++ к обфускации:
   a. Включите в него заголовочный файл omni_common.h с помощью директивы #include. Данный файл лежит в папке include, находящейся рядом с Obfuscator.exe. Его можно просто скопировать в проект; тогда директива #include будет выглядеть как #include "omni_common.h".
   b. Убедитесь, что код остался компилируемым и правильно работающим.
   c. Выберите функцию, которую вы собираетесь обфусцировать, и отметьте её специальным атрибутом, вставив перед определением функции строку

      ```
      __attribute__((annotate("StarForce::Obfuscator::Obfuscate(1)")))
      ```
5. В папке config, находящейся рядом с Obfuscator.exe, найдите конфигурационный файл, соответствующий вашему компилятору и целевой платформе. Предположим, вы разрабатываете ПО с помощью Visual Studio 2015 и компилируете программу под платформу x8664. Тогда вашим конфигурационным файлом будет targetConfig_VS2015_x64.ini. Обратите внимание, что в этом файле прописаны пути к вашим include-папкам (так как обфускатор должен знать, где искать включаемые файлы). Если используемый компилятор установлен не в папку по умолчанию, отредактируйте эти пути. Если вы используете какие-либо дополнительные библиотеки, добавьте их include-папки в конфигурационный файл. Если вы используете какие-либо дополнительные определения макросов или опции компилятора не по умолчанию, также добавьте их в конфигурационный файл (параметры Defines и Options соответственно).
6. Запустите обфускацию. Предположим, ваш исходный файл называется test.cpp, а обфусцированный файл должен называться test_obf.cpp. Тогда командная строка обфускатора будет выглядеть так:

   ```
   Obfuscator.exe test.cpp test_obf.cpp /TargetConfig=<путь_к_выбранному_конфигурационному_файлу>
   ```
7. Откройте полученный файл test_obf.cpp и ознакомьтесь с тем, как выглядит ваша функция после обфускации.
8. Скомпилируйте файл test_obf.cpp и убедитесь в работоспособности полученной программы.

3 Глоссарий

Термин	Описание
Обфускация	Запутывание кода программы, то есть приведение исходного кода или исполняемого кода к виду, сохраняющему функциональность программы, но затрудняющему анализ, понимание алгоритмов работы и их модификацию.
Анализ программы в статике	Анализ кода программы с использованием исполняемого файла или снимка исполняемого кода из памяти без исполнения этого кода. Обычно производится с использованием дизассемблера, например, IDA.
Анализ программы в динамике	Анализ кода и данных программы в момент её исполнения с использованием пошаговой трассировки и условных точек остановки. В отличие от статического анализа, при таком анализе учитывается состояние программы. Обычно производится с использованием отладчиков.
Анализ программы с использованием автоматических средств	Анализ кода и данных программы в момент её исполнения с использованием отладчика, который позволяет выполнять сценарии отладки и сохранять нужные значения во время отладки в журнал отладки.
Статистический анализ программы	Анализ с использованием сопоставления результатов работы программы в различных ситуациях с целью выяснения основных ветвей алгоритмов. Такой анализ позволяет отбросить все ветки и переходы, которые ни разу не выполнялись во время работы программы и, тем самым, произвести упрощение кода. Недостатком такого подхода является то, что могут быть не учтены некоторые переходы, и алгоритм уже не будет функционально-аналогичным.
Базовый компилятор	Компилятор C++ Clang 14.0.0, используемый для разбора входного файла для обфускации.
Целевой компилятор	Компилятор, используемый разработчиком.
Уровень обфускации	Относительное количество преобразований, произведённое над функцией или программой. Чем выше уровень, тем сложнее анализировать эту функцию, но при этом также возрастает размер полученной функции и увеличивается время её исполнения.

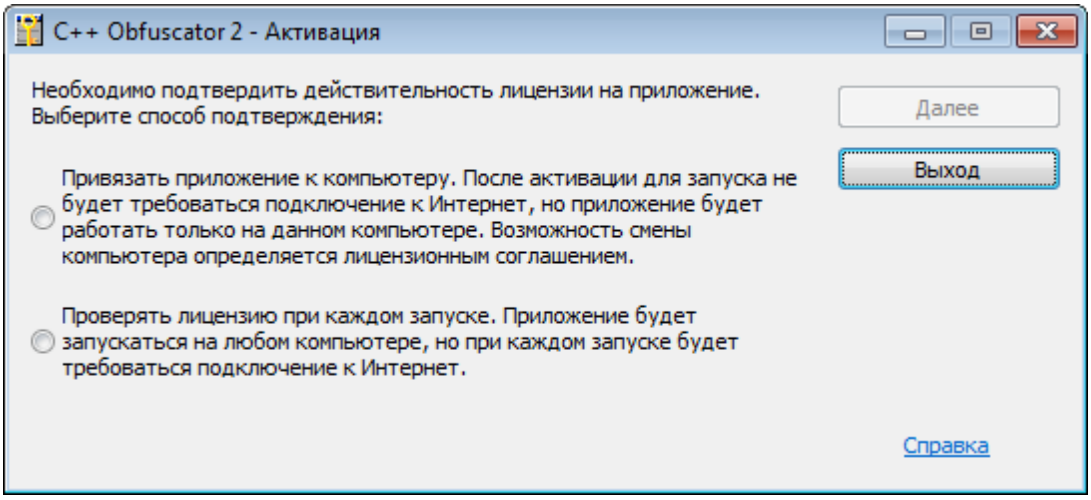
Глоссарий

4 Порядок использования

4.1 Активация StarForce C++ Obfuscator

Перед началом использования StarForce C++ Obfuscator необходимо активировать. Серийный номер для активации указан в письме, присланном вам при покупке программы.

При первом запуске StarForce C++ Obfuscator вы можете выбрать способ активации – привязать продукт к данному компьютеру или проверять лицензию при каждом запуске:



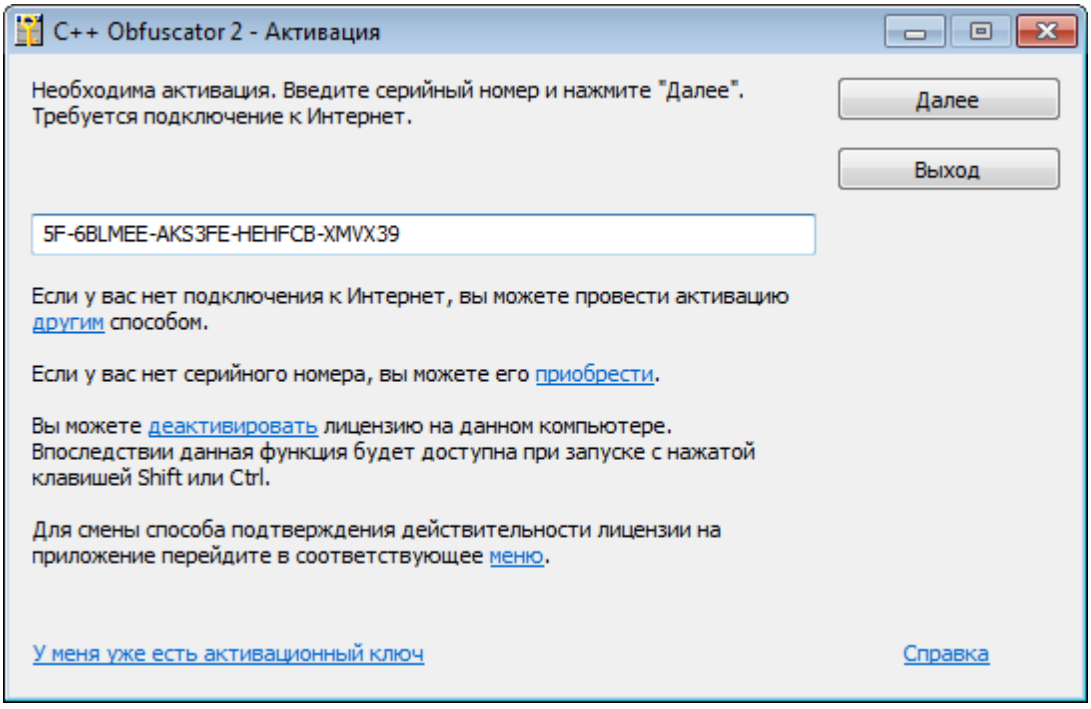
Выбор способа активации

Привязка к компьютеру позволяет запускать продукт только на данном компьютере, однако после активации подключение к интернету не требуется.

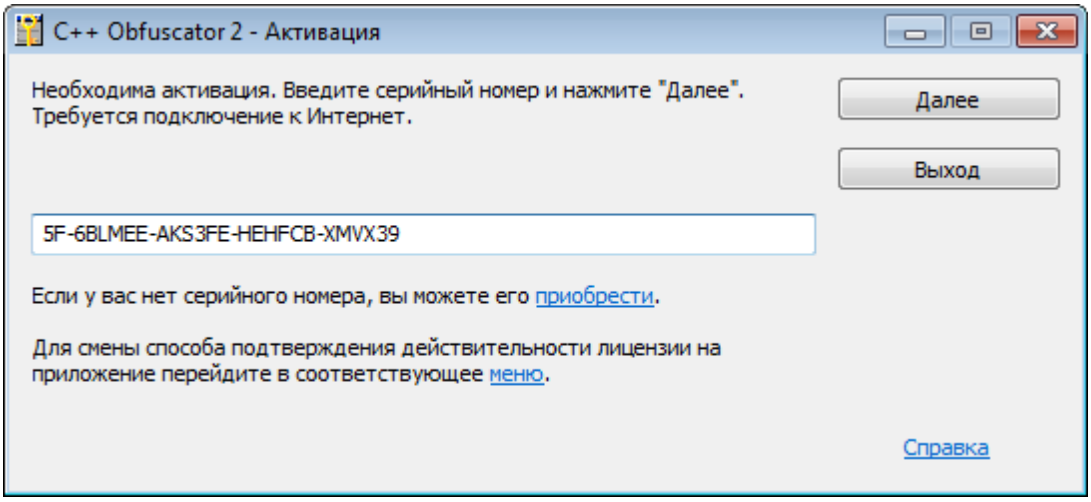
В случае проверки лицензии при каждом запуске продукт может запускаться на любом компьютере; при этом необходимо подключение к интернету. Менять компьютер, с которого запускается продукт, можно с установленной частотой (по умолчанию не ранее, чем через 30 мин).

Замечание. В случае работы под эмулятором Wine поддерживается только вариант проверки лицензии при каждом запуске.

В появившемся окне активации введите указанный в письме серийный номер и нажмите на кнопку **Далее**:



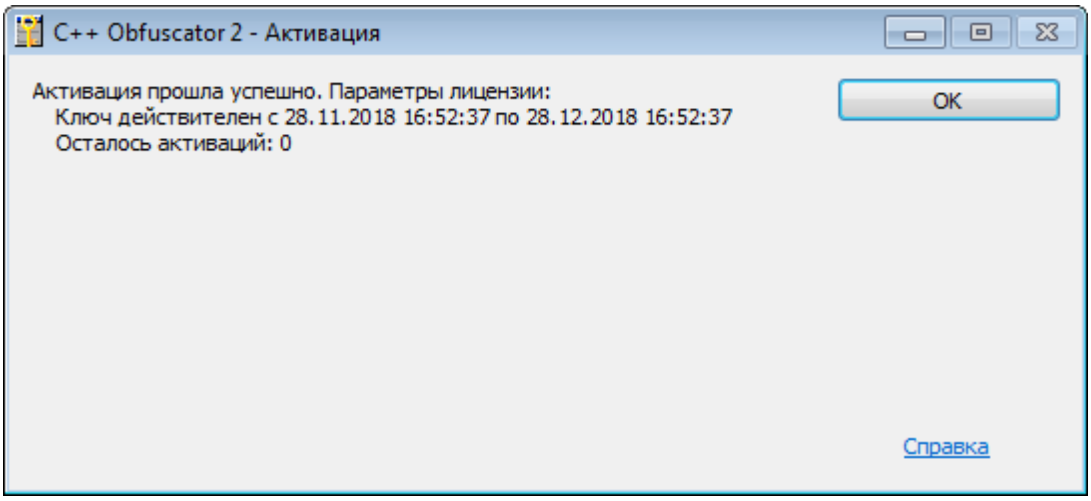
Окно активации с серийным номером (привязка к компьютеру)



Окно активации с серийным номером (проверка лицензии при каждом запуске)

Автоматическая активация

Продукт автоматически подключается к активационному серверу и, если серийный номер был введен верно, выполняется активация. В появившемся окне отображаются параметры лицензии:

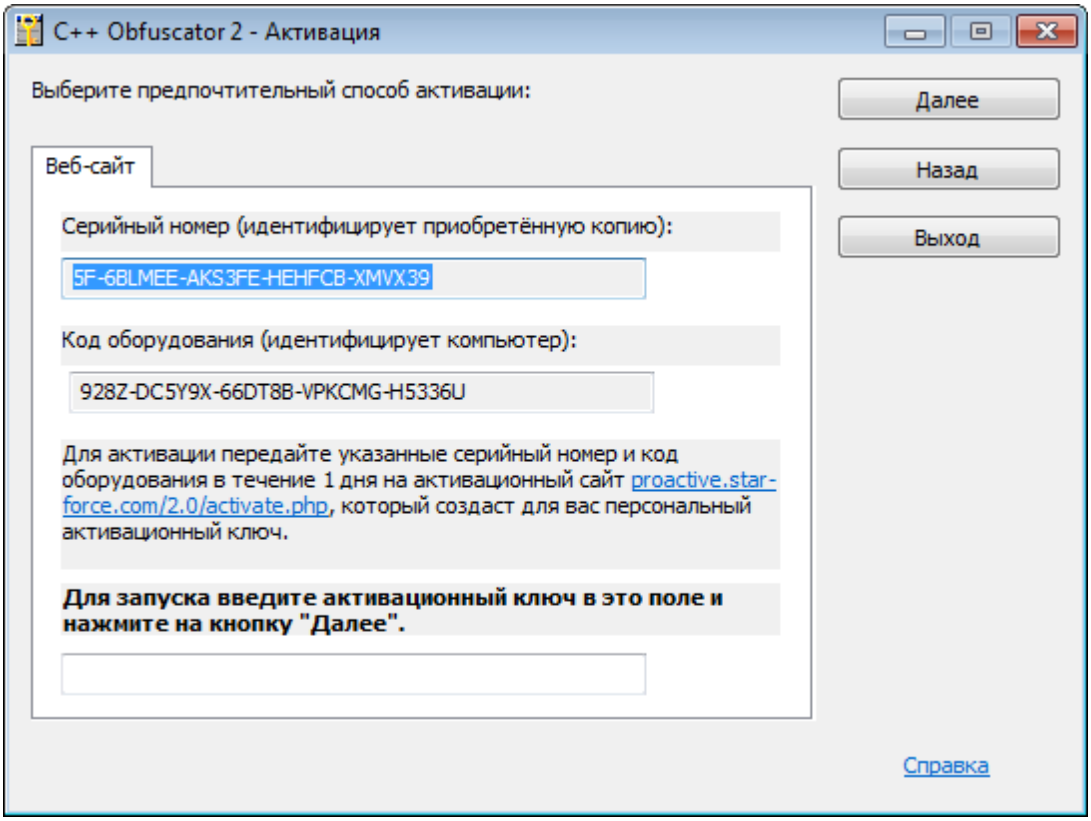


Продукт активирован

Нажмите на кнопку **OK**, чтобы завершить активацию StarForce C++ Obfuscator.

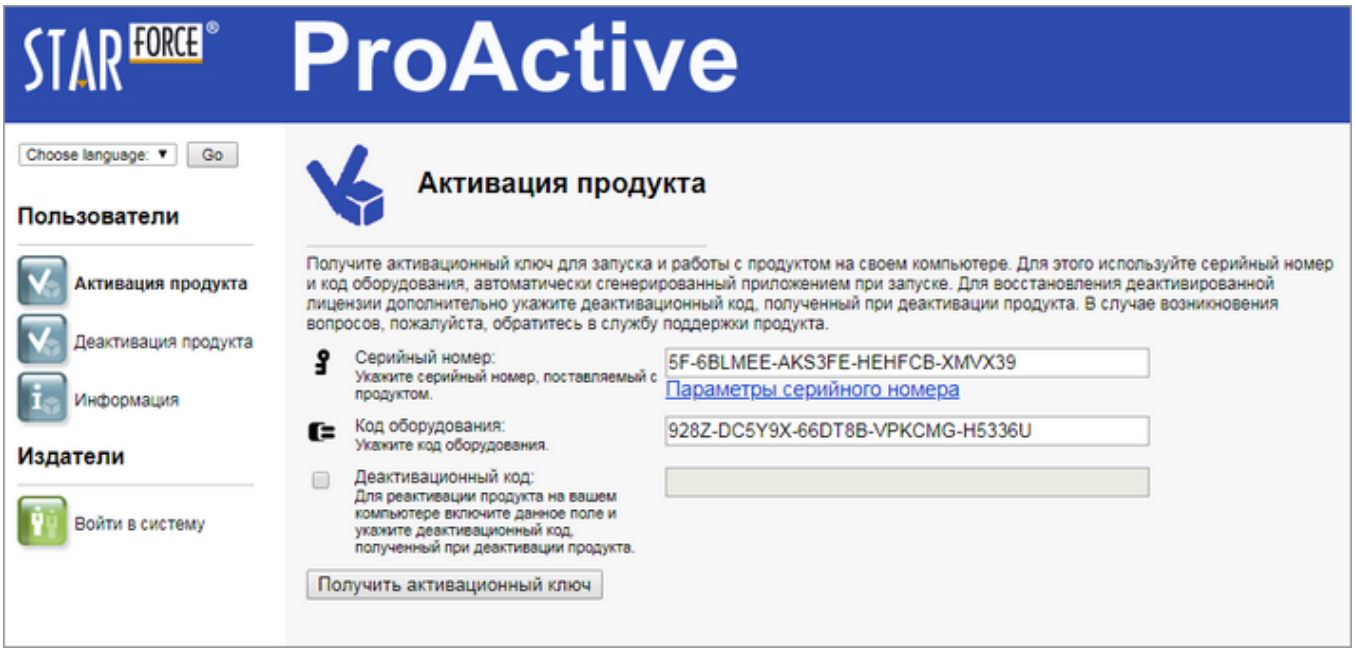
Ручная активация на компьютере с подключением к Интернету (только для привязки к компьютеру)

Данный тип активации можно использовать в случае, если по каким-то причинам (из-за брандмауэра или настроек сети) автоматическая активация не срабатывает. Введите серийный номер в окне активации (см. выше) и нажмите на ссылку **другим**. Скопируйте серийный номер и код оборудования, указанные в появившемся окне:



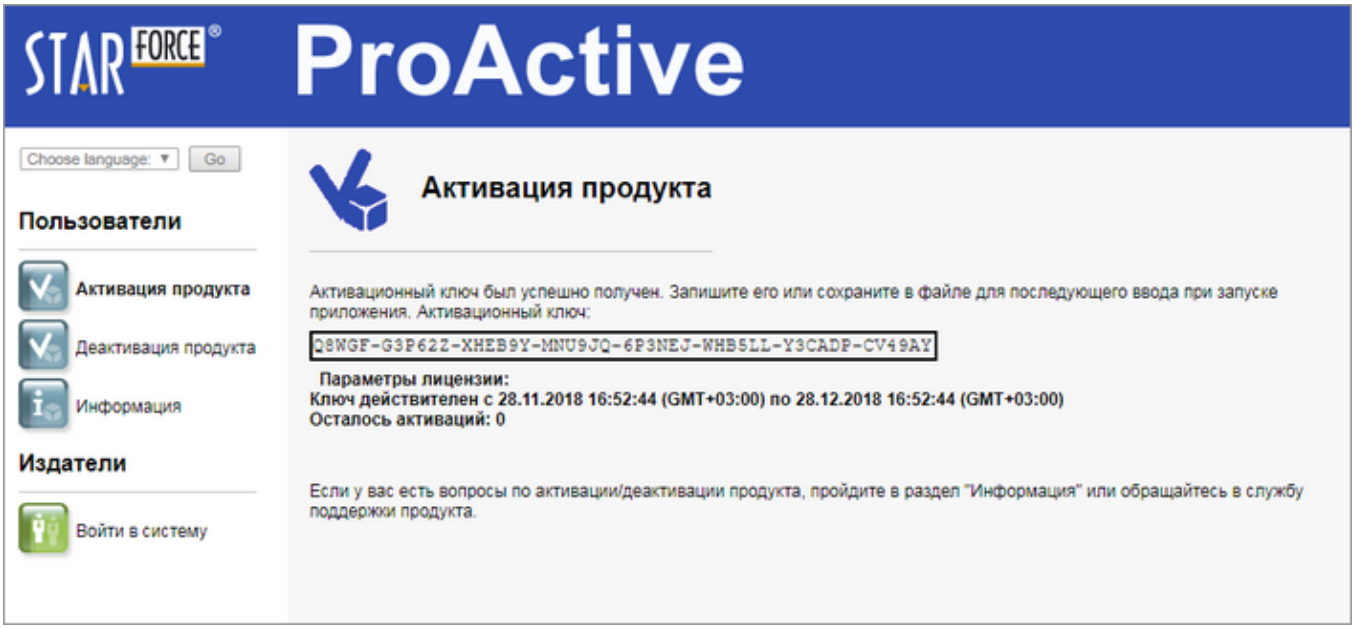
Активация вручную

Перейдите в раздел **Активация продукта** на сайте StarForce ProActive, введите серийный номер и код оборудования в соответствующие поля и нажмите на кнопку **Получить активационный ключ**:



Активация на сайте ProActive

Скопируйте сгенерированный активационный ключ:



Получение активационного ключа

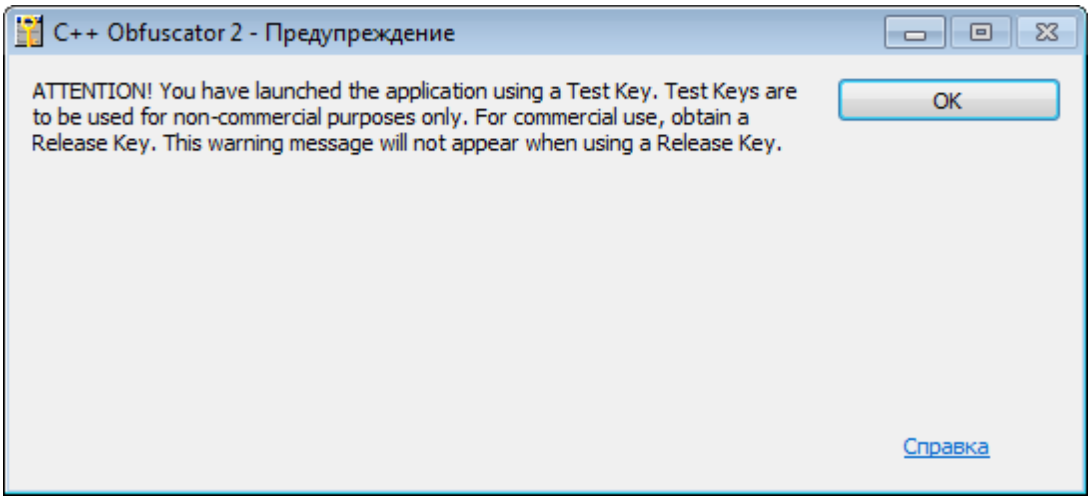
Введите активационный ключ в соответствующее поле в окне активации и нажмите на кнопку **Далее**. После этого процесс активации StarForce C++ Obfuscator завершается.

Замечание. Система защиты запоминает выбранный при первом запуске StarForce C++ Obfuscator способ активации. Вы можете изменить его, запустив файл Omniform.exe и удерживая клавишу **Shift**; при этом отображается окно выбора типа активации.

Активация тестовым серийным номером

По умолчанию используемый для активации серийный номер является тестовым и имеет следующие ограничения:

- Максимальный размер обфусцируемого файла – 10 000 байтов.
- Срок действия серийного номера – 15 дней.
- При каждом запуске StarForce C++ Obfuscator выводится сообщение о том, что продукт запущен с помощью тестового ключа:



Предупреждение об использовании тестового серийного номера

Чтобы использовать StarForce C++ Obfuscator без ограничений, обратитесь к вашему менеджеру для получения релизного серийного номера. Активируйте продукт ещё раз, используя полученный

серийный номер. Активация выполняется аналогично.

4.2 Настройка обфускатора для работы с целевым компилятором

Для выполнения своей задачи обфускатор должен самостоятельно произвести лексический и синтаксический анализ исходного кода обфусцируемого файла (в том числе всех включаемых заголовочных файлов), то есть выполнить часть работы компилятора. Для этой цели служит специальный логический модуль обфускатора – базовый компилятор. В качестве базового компилятора StarForce C++ Obfuscator 2.0 используется компилятор Clang 14.0.0. Этот компилятор уже встроен в Obfuscator.exe и не требует дополнительной установки.

Несмотря на то, что язык C++ стандартизован, каждая программа, написанная на C++, ориентирована, как правило, на использование конкретного компилятора и на компиляцию под конкретную целевую платформу. К числу программных конструкций, создающих такую привязку к компилятору и целевой платформе, относятся, например, следующие:

1. Размеры стандартных типов.
2. Особенности реализации стандартной библиотеки.
3. Языковые конструкции, специфичные для данного компилятора.

После использования обфускатора привязка к целевому компилятору и платформе становится ещё более выраженной. Пусть, например, обфускатор для запутывания кода преобразовывает переменную типа void* в массив байтов, далее работает с этими байтами по отдельности, а в конце концов собирает их снова в единое целое. Очевидно, что для выполнения подобных преобразований обфускатор должен точно знать размер указателя (обычно 32 или 64 байта), и что обфусцированный код не будет работать на платформе с другим размером указателя.

В связи с этим при обфускации кода необходимо:

1. Сообщить обфускатору, с какой платформой и каким компилятором ему придётся работать. После обфускации код можно будет компилировать только с указанной конфигурацией. Если исходная программа компилируется в двух вариантах (например, под платформы x8632 и x8664), то и обфусцироваться она должна дважды, отдельно под каждую из платформ.
2. Сообщить обфускатору пути к стандартным include-папкам, чтобы он смог найти нужные заголовочные файлы.
3. Сообщить обфускатору тот же набор заранее заданных определений, которые задаются компилятору при компиляции данного файла. Типичным определением, указываемым не в программе, а в параметрах компилятора, является указание на сборку в отладочном режиме (аналог определения #define DEBUG).

Вся эта информация передаётся обфускатору через конфигурационный файл параметров компилятора, задаваемый в командной строке обфускатора параметром /TargetConfig. Этот файл содержит следующие настройки:

- Defines – перечисление определений (через точку с запятой).
- Includes – пути к каталогам с заголовочными файлами (через точку с запятой).
- Options – опции базового компилятора (через пробел). Наиболее важной опцией является -target, задающая целевую платформу и целевой компилятор (например, i386-pc-windows-msvc).

В комплекте с обфускатором в папке config находится набор заранее подготовленных конфигурационных файлов параметров обфускатора для наиболее распространённых компиляторов. Кроме того, такой файл можно подготовить самостоятельно, пользуясь описанием из раздела Конфигурационный файл параметров компилятора (TargetConfig).

Дополнительные опции базового компилятора можно также задать в командной строке, см. раздел Запуск обфускатора.

4.3 Выбор функций для обфускации

4.3.1 Подготовка функции для обфускации

1. Каждый исходный файл, содержащий функции для обфускации, должен включать заголовочный файл omni_common.h из комплекта поставки обфускатора. Для удобства включения вы можете скопировать этот файл из папки с обфускатором в любую папку на компьютере.
2. Проверьте, что каждая обфусцируемая функция не использует неподдерживаемых обфускатором конструкций, и, если такие конструкции есть в функции, видоизмените её, либо выберите для обфускации другую функцию (см. раздел Ограничения на выбор функций для обфускации).
3. Для обфускации функции задайте для неё атрибут annotate("StarForce::Obfuscator::Obfuscate(<уровень_обфускации>)"), где уровень_обфускации – целое число (от 0 – не обфусцировать до 5 – максимальный уровень), например, перед описанием функции на той же или предыдущей строке.

   Пример:

```
__attribute__((annotate("StarForce::Obfuscator::Obfuscate(2)")))
void foo()
{
....
}
```

   Значение 0 уровня обфускации означает отсутствие обфускации. В этом случае функция будет пропущена, и в консоль будет выведено соответствующее сообщение.
4. Как правило, обфусцированный код при компиляции порождает большое количество предупреждений (warnings). Для удобства работы рекомендуется отключить их. Набор предупреждений и директивы для их отключения специфичны для компилятора. В частности, почти все компиляторы выдают предупреждения о неиспользуемых переменных и метках, на которые нет переходов.

4.3.2 Рекомендации по выбору функций для обфускации

Во избежание проблем с работой обфусцированной программы соблюдайте следующие рекомендации:

1. Не пытайтесь обфусцировать все функции подряд, так как это может существенно увеличить размер получаемого кода и замедлить процесс компиляции (или даже привести к падению некоторых компиляторов). Вместо этого выберите функции, выполняющие действия, которые вы действительно хотите скрыть.
2. Функции должны быть некритичны к скорости выполнения. При выборе сильного уровня обфускации время выполнения функции может быть увеличено в десятки раз.
3. Учитывайте ограничения на выбор функций для обфускации.

Для обеспечения хорошего уровня защиты соблюдайте следующие рекомендации:

1. Функций должно быть достаточно много.
2. Желательно, чтобы функции содержали большое количество ветвлений (таких, как операторы if, for и do/while). Часть ветвлений должна выполняться при определенных условиях, тогда задача анализа будет гораздо сложнее (но эти ветки должны быть необходимы для работы программы).
3. Не рекомендуется обфусцировать простые функции с низким уровнем обфускации, так как в этом случае обфусцируемый код может оказаться эквивалентным необфусцированному после

оптимизации компилятором.

4.3.3 Ограничения на выбор функций для обфускации

В таблице ниже перечислены ограничения, накладываемые на функции для обфускации, в зависимости от используемых элементов и конструкций.

Элемент, конструкция	Ограничение
Windows SEH	Нельзя обфусцировать функции, использующие Windows SEH (__try/__exept, __try/__finally). В обфусцируемых функциях допустимы лишь стандартные обработчики исключений C++ (try/catch).
Ассемблерные вставки	Нельзя обфусцировать функции, содержащие ассемблерные вставки.
Конструкторы и деструкторы	Нельзя обфусцировать конструкторы и деструкторы. При необходимости обфускации конструкторов и деструкторов вынесите их тела в отдельные обфусцируемые функции, которые будут вызываться из необфусцируемых конструкторов и деструкторов.
Локальные классы, типы и прототипы функций	В обфусцированных функциях нельзя описывать локальные классы, определять типы (typedef) и прототипы внешних функций. Вынесите все подобные определения из тела функции.
Инициализация структур с константными полями	В обфусцированных функциях нельзя инициализировать структуры, классы, объединения, если среди их членов есть ссылки или поля, описанные как константы или ссылки, например, const int member.
Лишние вызовы копирующих и перемещающих конструкторов	Нельзя обфусцировать функции, использующие объекты, для которых недопустимы лишние вызовы копирующих или перемещающих конструкторов. Это связано с тем, что в некоторых случаях обфускатор может генерировать лишние вызовы копирующих или перемещающих конструкторов. Типовые случаи, при которых могут сгенерироваться такие вызовы: • Порождение исключения (для объекта исключения). • Возврат из функции экземпляра объекта. • Передача параметра, являющегося экземпляром объекта, в вызываемую функцию.
Неименованные структуры	В обфусцируемых функциях нельзя обращаться к полям или вызывать методы неименованных структур.
Вычисляемые значения статических переменных	В обфусцируемых функциях не рекомендуется использовать статические переменные, инициализируемые вычисляемым значением (не константой), так как их инициализация будет производиться без критической секции.
Статические переменные, определяемые в теле функции	В обфусцируемых функциях нельзя объявлять статические переменные, так как для них неправильно выполняется инициализация. При необходимости использовать статические переменные вынесите их из функции и сделайте глобальными переменными (или статическими членами класса, если обфусцируемая функция – это метод класса).
#pragma-директивы	Внутри обфусцируемых функций не рекомендуется использовать #pragma-директивы, так как они не записываются в выходной файл. Вынесите все #pragma-директивы из обфусцируемых функций.
Дополнительные спецификаторы функций	Если в определении обфусцируемой функции используются дополнительный спецификатор extern "C" (language linkage specification), он должен стоять перед атрибутом, задающим обфускацию функции.
Default-параметры	Если в определении функции используются default-параметры, то они должны задаваться в более раннем (по коду) объявлении (declaration) функции.

Ограничения для обфусцируемых функций

4.4 Запуск обфускатора

StarForce C++ Obfuscator предполагает обфускацию одного файла без запуска целевого компилятора. Командная строка обфускатора имеет следующий вид:

```
obfuscator <InputFile> <OutputFile> [Obfuscator Options] [Additional Options]
```

где

InputFile – имя входного файла;

OutputFile – имя результирующего файла;

Obfuscator Options:

/TargetConfig=FileName – конфигурационный файл параметров компилятора. Значение опции по умолчанию – config/targetConfig.ini. Если указан относительный путь к файлу (как, например, для значения по умолчанию), то обфускатор ищет файл сначала относительно текущей папки, а затем относительно Obfuscator.exe (поэтому файл по умолчанию обфускатор находит всегда). В дистрибутиве в папке config поставляется набор готовых файлов для обфускации под разные платформы и компиляторы. Типовой синтаксис для указания готового файла – /TargetConfig=config\<имя_файла>. Назначение конфигурационного файла параметров компилятора описано в разделе Настройка обфускатора для работы с целевым компилятором; перечень возможных параметров – в разделе Конфигурационный файл параметров компилятора (TargetConfig).

/ObfuscationConfig=FileName – конфигурационный файл параметров обфускатора. Значение опции по умолчанию – \config\standard.ini. Алгоритм поиска файла такой же, как для /TargetConfig. Как правило, конфигурационного файла по умолчанию достаточно для решения практических задач, поэтому задавать данную опцию требуется редко. Перечень возможных полей, задаваемых в конфигурационном файле параметров обфускатора, приведён в разделе Конфигурационный файл параметров обфускатора (ObfuscationConfig).

/LogFile=FileName – имя log-файла. По умолчанию создается файл с именем выходного файла и приписанным расширением .log.

/PreserveLog – не удалять log-файл в случае успешного завершения процесса обфускации. По умолчанию после успешной обфускации log-файл не сохраняется.

Additional Options – дополнительные опции базового компилятора (Clang 14.0.0), которые могут содержать список дополнительных каталогов Include (аналогично -I или /I компилятора) и дополнительные определения для препроцессора (аналогично -D или /D компилятора).

Нестандартные каталоги Include должны указываться с помощью ключей –I или /I; нестандартные определения для препроцессора – с помощью –D или /D.

Наиболее важные опции Clang 14.0.0 описаны с разделе Конфигурационный файл параметров компилятора (TargetConfig). Полный список опций см. здесь.

Пример:

```
obfuscator Sample.cpp Obfuscated\Sample.cpp /TargetConfig=targetConfig_VS2017_x64.ini
```

4.5 Встраивание обфускации в процесс сборки проекта

Для сборки обфусцированной программы целевому компилятору нужно передать обфусцированные файлы вместо необфусцированных.

Для удобства работы необходимо, чтобы это делалось автоматически, и чтобы для целей отладки сохранялась возможность сборки необфусцированного файла. Таким образом, возникает задача встраивания обфускатора в процесс сборки проекта. Вы можете решить эту

задачу, например, следующими способами:

1. Если проект использует для сборки make-файл, то задача встраивания решается созданием отдельной конфигурации, которая использует обфускацию. В этой конфигурации присутствует вызов обфускатора, а в результирующую сборку просто включаются обфусцированные файлы вместо необфусцированных.
2. Для любых проектов можно также использовать директивы препроцессора для компиляции исходного или обфусцированного файла. Пусть, например, имеется необфусцированный файл source.cpp, а перед началом сборки (либо перед началом компиляции данного файла) вызывается обфускатор, создающий файл source_obf.cpp. Оформим source.cpp следующим способом:

```
// source.cpp
#include "omni_common.h"
#ifdef USE_OBFUSCATOR
#include "source_obf.cpp"
#else
// Необфусцированный код source.cpp
...
#endif
```

   В этом случае, если задано определение USE_OBFUSCATOR, то будет скомпилирована обфусцированная программа; если нет – необфусцированная. Очевидно, что для успешной обфускации обфускатор не должен пойти по первому сценарию, поэтому параметр USE_OBFUSCATOR ему в командной строке передавать не нужно.

Замечания.

1. Обфусцированный код зависит от платформы, и для каждой платформы его необходимо обфусцировать заново. Чтобы предотвратить сбои в работе приложения, никогда не используйте в одном проекте файлы, обфусцированные для разных платформ.
2. В обфускатор следует передавать те же определения, которые передаются в целевой компилятор (кроме рассмотренного в пункте 2 определения USE_OBFUSCATOR). Для этого можно использовать параметр командной строки обфускатора /D, либо задать нужные определения в конфигурационном файле параметров компилятора (TargetConfig).

4.6 Отладка обфусцированных файлов

Если обфусцированный файл не работает, причины могут быть следующими:

1. Не работает также необфусцированный файл. При анализе проблемы исключите эту возможную причину в первую очередь, проверив работоспособность необфусцированного файла.
2. Неправильно работает обработка кода базовым компилятором, либо генератором исходного текста. Для проверки этого предположения выполните обфускацию с параметром Enabled = 0, задаваемом в конфигурационном файле параметров обфускатора (см. раздел Конфигурационный файл параметров обфускатора (ObfuscationConfig)). В этом случае в выходном файле сгенерируется код, пропущенный через обфускатор, но без применения методов обфускации. Если данный код работает, то причина, вероятно, в самой обфускации. Если код не работает, то причина именно в неправильной работе базового компилятора или генератора исходного текста.

   Для дальнейшего анализа и решения проблемы рассмотрите следующие подходы:

   a) Проверьте правильность задания параметров базового компилятора в конфигурационном файле параметров компилятора (TargetConfig). Проверить правильность задания параметров помогает указание в конфигурационном файле параметров компилятора опций Clang -H для проверки корректности указания включаемых файлов и -v для вывода более подробной информации.

b) Локализуйте функцию, из-за которой происходит проблема, отключая преобразование отдельных функций или файлов.

c) Проверьте, не встречается ли в проблемной функции какая-либо не поддерживаемая обфускатором конструкция (см. Ограничения на выбор функций для обфускации).

d) Проанализируйте преобразованную обфускатором найденную проблемную функцию. Если обфускация отключена, то сделать это довольно просто. Возможно, функцию можно переписать, чтобы исключить проблемное место.

e) Отлаживайте работу файла методом промежуточной печати.

f) Обратитесь в службу технической поддержки.

3. Неправильно работает сама обфускация. Для решения можно использовать следующие подходы:

a) Локализуйте функцию, из-за которой происходит проблема, устанавливая для функций нулевой уровень обфускации.

b) Попробуйте снизить уровень обфускации для проблемной функции, или изменить параметры обфускации с помощью конфигурационного файла параметров обфускации.

c) Отлаживайте работу файла методом промежуточной печати.

d) Обратитесь в службу технической поддержки.

Замечание. При обращении в службу технической поддержки StarForce передайте следующую информацию:

- log-файл;
- полное описание среды;
- используемый файл targetConfig.ini;
- дополнительные библиотеки (если используются) и пути к ним;
- исходный файл.

5 Справочные материалы

5.1 Состав дистрибутива

StarForce C++ Obfuscator состоит из следующих компонентов:

- Исполняемый файл StarForce C++ Obfuscator (Obfuscator.exe).
- Папка config, содержащая стандартные настройки обфускации (файл standart.ini; см. раздел Общие настройки конфигурационного файла) и конфигурационные файлы для разных платформ и компиляторов (targetConfig.<platform>.ini), например:

  targetConfig_VS2013_x64.ini,

  где VS2013 – версия компилятора, x64 – разрядность ОС.
- Папка include, содержащая единственный заголовочный файл обфускатора omni_common.h для включения в обфусцируемые исходные коды.
- Папка doc, содержащая руководство пользователя.

5.2 Системные требования

1. StarForce C++ Obfuscator работает под следующими ОС Windows:
   - Windows 7 x8632/x8664 Professional, Ultimate, Enterprise;
   - Windows 8 x8632/x8664 Pro, Enterprise;
   - Windows 8.1 x8632/x8664 Pro, Enterprise;
   - Windows 10 x8632/x8664 Pro, Enterprise;
   - Windows Server 2008 R2 x8664 (все ревизии);
   - Windows Server 2012 x8664 (все ревизии);
   - Windows Server 2012 R2 x8664 (все ревизии);
   - Windows Server 2016 x8664 (все ревизии).
2. StarForce C++ Obfuscator работает под эмулятором Wine 2.0.1 и 3.0 на следующих базовых ОС:
   - Ubuntu 16.04 LTS x8632;
   - Ubuntu 18.04 LTS x8664;
   - macOS 10.12.6.
3. Минимальные системные требования:
   - Объём ОЗУ – 2 Гб;
   - Объём свободного пространства на жёстком диске – 10 Гб.

5.3 Поддерживаемые компиляторы и платформы

StarForce C++ Obfuscator поддерживает большое количество компиляторов. Ниже приведён список компиляторов, на которых StarForce C++ Obfuscator тестировался. Поддержка других компиляторов не гарантируется (данные о совместимости или несовместимости с ними отсутствуют).

Поддерживаемые компиляторы:

1. Для платформы Windows x8632/x8664:
   - MSCV++ 8.0, _MSC_VER == 1400 (Microsoft Visual Studio 2005);
   - MSVC++ 11.0, _MSC_VER == 1700 (Microsoft Visual Studio 2012 version 11.0);
   - MSVC++ 14.16, _MSC_VER == 1916 (Microsoft Visual Studio 2017 version 15.9);
   - Clang 3.8.1;
   - Clang 3.9.1;
   - Clang 4.0.1;
   - Clang 5.0.2;
   - Clang 6.0.1;
   - Clang 7.0.0.
2. Для платформы Linux x8632/x8664/ARM32/ARM64:
   - GCC 5.5;

- GCC 6.4;
- GCC 7.3;
- GCC 8.2.

3. Для платформы macOS x8664:
   - Apple LLVM 8.0.0 (Xcode 8.0-8.2);
   - Apple LLVM 8.1.0 (Xcode 8.3-8.3.3);
   - Apple LLVM 9.0.0 (Xcode 9.0-9.2);
   - Apple LLVM 9.1.0 (Xcode 9.3-9.4);
   - Apple LLVM 10.0.0 (Xcode 10.0).
4. Для платформы Android ARM32/ARM64:
   - Clang 3.6 (Android NDK, r10e);
   - Clang 3.8 (Android NDK, r11-r18);
   - GCC 4.6 (Android NDK, r8b-r10d);
   - GCC 4.7 (Android NDK, r8e-r10d);
   - GCC 4.8/4.8.3 (Android NDK, r9-r10e);
   - GCC 4.9 (Android NDK, r11-r12b).
5. Для платформы iOS ARM32/ARM64:
   - Apple LLVM 10.0.1 (Xcode 10.2).

Неподдерживаемые компиляторы:

- Microsoft Visual Studio 2017 15.0 (RTM).

5.4 Конфигурационный файл параметров компилятора (TargetConfig)

Конфигурационный файл параметров компилятора позволяет добиться одинакового понимания исходного кода обфускатором и целевым компилятором. В нём задаются, в частности, такие параметры, как пути поиска заголовочных файлов и целевая платформа.

Более подробно о проблемах, связанных с обеспечением совместимости обфускатора и целевого компилятора, см. в разделе Настройка обфускатора для работы с целевым компилятором.

В данном разделе представлена справочная информация о параметрах конфигурационного файла, а также описаны особенности работы с конфигурационным файлом, если на компьютере установлено несколько компиляторов.

5.4.1 Параметры конфигурационного файла

Следующий пример даёт представление о синтаксисе конфигурационного файла параметров компилятора:

```
[Settings]
; Configuration file for Microsoft Visual Studio 2017 (target platform x86-32)
Defines = WIN32;NDEBUG;_CONSOLE;_UNICODE;UNICODE
Includes = C:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise\VC\Tools\MSVC\14.16.27023\include;C:
\Program Files (x86)\Windows Kits\10\include\10.0.10240.0\ucrt
Options = -target=i386-pc-windows-msvc -std=c++17 -fms-extensions -fms-compatibility -fms-compatibility-
```

version=19.1 -fcxx-exceptions

Синтаксис конфигурационного файла следующий:

- Файл начинается со строки [Settings].
- Файл может содержать произвольное число комментариев, начинающихся с точки с запятой.
- Файл может задавать три параметра (Defines, Includes и Options). Каждый из параметров может отсутствовать; в этом случае базовому компилятору соответствующая информация не передаётся.
- Параметр Defines задаёт набор определений, передаваемых базовому компилятору (через точку с запятой). Указание некоторого параметра DEF1 без значения эквивалентно директиве #define DEF1 1 в исходном коде. Указание некоторого параметра DEF2=VAL со значением эквивалентно директиве #define DEV2 VAL в исходном коде.
- Параметр Includes задаёт пути к каталогам с заголовочными файлами (через точку с запятой).
- Параметр Options задаёт опции базового компилятора (через пробел). Наиболее важной опцией является -target, задающая целевую платформу и целевой компилятор (например, i386-pc-windows-msvc).

Полный список опций базового компилятора описан в документации на базовый компилятор. Наиболее важные опции описаны в таблице ниже.

Параметр	Описание	Рекомендации по использованию
-target <triple>	Общий формат <triple> следующий: <arch><sub>-<vendor>-<sys>-<abi>, где: arch = x86_64, i386, arm, thumb, mips, etc. sub = for ex. on ARM: v5, v6m, v7a, v7m, etc. vendor = pc, apple, nvidia, ibm, etc. sys = none, linux, win32, darwin, cuda, etc. abi = eabi, gnu, android, macho, elf, etc.	Наиболее важный параметр; указывает целевую программно-аппаратную платформу и компилятор (ABI компилятора).
-std=<arg>	Используемый языковой стандарт. Примеры допустимых значений: c89, c99, c11, c++03, c++11, c++14 (по умолчанию), c++17.	В обфускаторе должен использоваться тот же стандарт, что и в целевом компиляторе.
-H	Выводить на консоль используемые включаемые файлы.	Рекомендуется использовать для отладки настроек путей к включаемым файлам (обфускатор и целевой компилятор должны использовать одни и те же файлы).
-v	Выводить на консоль детальную информацию о процессе обработки исходного файла.	Рекомендуется использовать для отладки.
-fexceptions, -fno-exceptions	Включить/отключить поддержку исключений.	Обработка исключений в обфускаторе должна быть настроена так же, как и в целевом компиляторе.
-fcxx-exceptions, -fno-cxx-exceptions	Включить/отключить поддержку исключений C++.	
-fdwarf-exceptions	Использовать механизм обработки исключений DWARF.	
-fseh-exceptions	Использовать механизм обработки исключений SEH.	
-fsjlj-exceptions	Использовать механизм обработки исключений setjmp/longjmp.	

Основные опции базового компилятора

5.4.2 Если на компьютере установлено несколько компиляторов

Если на компьютере установлено несколько компиляторов (особенно в случае, если установлено несколько версий одного компилятора), при использовании обфускатора следует учитывать следующие обстоятельства:

- Каждый компилятор, как правило, поставляется с собственным набором заголовочных файлов.
- Компиляторы могут быть установлены не в папки по умолчанию.

Это может привести к тому, что поставляемые вместе с обфускатором конфигурационные файлы параметров компилятора не будут корректно указывать пути к заголовочным файлам.

Поэтому в случае наличия на компьютере нескольких компиляторов необходимо особенно тщательно следить за тем, чтобы целевой компилятор и обфускатор использовали одни и те же заголовочные файлы. Чтобы проверить пути к заголовочным файлам, реально используемым обфускатором, вы можете указать в командной строке обфускатора дополнительный параметр -H.

5.5 Конфигурационный файл параметров обфускатора (ObfuscationConfig)

Параметры алгоритмов обфускации настраиваются в отдельном конфигурационном файле. Этот файл состоит из раздела с общими настройками и с настройками, специфичными для каждого уровня обфускации. Параметры, указывающие вероятности, задаются значениями в процентах от 0 до 100. Уменьшение большинства вероятностей позволяет уменьшить размер кода и увеличить скорость его выполнения.

Как правило, вам не нужно менять имеющиеся значения, если только вы не хотите настроить параметры обфускации или устранить конкретные проблемы.

5.5.1 Общие настройки конфигурационного файла

Параметр	Описание	Рекомендации по использованию
Enabled	Разрешить обфускацию.	Отключение данной опции может потребоваться для отладки. Однако функции все равно будут преобразованы, хотя обфускация и не будет осуществляться (отлаживать в отладчике будет сложнее, так как исходный код будет существенно менее понятным).
RandomSeed	32-разрядное целое неотрицательное число, являющееся начальным значением для инициализации генератора случайных чисел. Если оно не задано, генератор случайных чисел инициализируется текущим временем.	Используется в случае, когда необходимо каждый раз получать одинаковый код на выходе при условии, что другие настройки не менялись.
UseStateVariables	Разрешить использование переменных состояния при обфускации.	Рекомендуется всегда оставлять включённым. Переменные состояния являются одним из базовых методов данной реализации обфускации и используются другими методами. Отключение может существенно снизить уровень обфускации.
UseFlowGraph	Разрешить использование графа потоков данных при обфускации.	Рекомендуется всегда оставлять включённым. Граф потоков данных используется многими методами, и его отключение может существенно снизить уровень обфускации.
MakeArgumentShadows	Включить создание копий аргументов в функциях.	Рекомендуется всегда оставлять включённым. Отключение данной опции может существенно снизить уровень обфускации.
StateVari	Количество переменных состояния,	Рекомендуемое значение – 8.

Параметр	Описание	Рекомендации по использованию
ableCount	используемых при генерации кода.	Может быть уменьшено для небольших функций или увеличено для очень больших. Рекомендуемый диапазон значений – от 4 до 16.

Общие настройки обфускации

Нижеприведенные настройки действуют на функцию, которая получается в процессе выделения базовых блоков из разных функций в общую функцию.

Параметр	Описание	Рекомендации по использованию
OutlinedFunctionLevel	Уровень обфускации для функции, которая получается в процессе выделения базовых блоков из разных функций в общую функцию.	Рекомендуется выбирать уровень, в котором параметры обфускации имеют достаточно большие значения.

Настройки для функции

Нижеприведенные настройки являются настройками виртуальной машины:

Параметр	Описание	Рекомендации по использованию
GenerateVM	Разрешить генерацию виртуальной машины.	Рекомендуется оставить включённым.
VMIPCount	Количество виртуальных указателей для инструкции виртуальной машины.	Рекомендуемое значение – 3. Увеличение этого значения увеличит скорость выполнения, а уменьшение не рекомендуется.
MinimumPatternSizeForVM	Минимальный набор последовательностей инструкций для команды виртуальной машины.	Рекомендуемое значение – 2. Увеличение уменьшит число инструкций. Не рекомедуется задавать значения, большие, чем 3.
MaximumPatternSizeForVM	Максимальный набор последовательностей инструкций для команды виртуальной машины.	Рекомендуемое значение – 6. Если задать значение меньше 6, то скорость создания виртуальной машины увеличится, но скорость исполнения обфусцируемой программы может уменьшиться.
InitializeVMIPProbability	Вероятность инициализации виртуальных указателей в тех блоках, где в них нет необходимости.	Рекомендуемое значение – 20. При увеличении данного значения увеличивается уровень обфускации, но также увеличивается размер кода и время его исполнения.
UseDispatcherIDForVMIPInitialization	Вероятность использования индекса диспетчера для инициализации виртуальных указателей.	Рекомендуемое значение – 10. Слишком большое значение может существенно увеличить размер кода.
GenerateAssertOnVMIPProbability	Вероятность использования виртуального указателя для генерации ложных переходов.	Рекомендуемое значение – 10. Слишком большое значение может существенно увеличить размер кода.
MinimumLevel	Значение минимального уровня обфускации, описанного в файле.	-
MaximumLevel	Значение максимального уровня обфускации, описанного в файле.	-

Настройки виртуальной машины

5.5.2 Настройки для каждого уровня обфускации

Параметр	Описание	Рекомендации по использованию
InliningLevel	Отношение количества инструкций в текущей функции к количеству инструкций в вызываемой функции, при котором производится подстановка тела функции на место её вызова. Чем больше параметр, тем меньше функций подставляется.	Рекомендуемое значение – 10. Параметр может быть увеличен для уменьшения размера кода (это повлечёт за собой уменьшение уровня обфускации) и уменьшен для улучшения уровня обфускации (это может повлечь за собой существенное

Параметр	Описание	Рекомендации по использованию
		увеличение размера кода).
TransformCallsProbability	Вероятность замены вызова функций по имени на вызов функций по адресу.	Рекомендуемое значение – 0 или 100. Если в обфусцируемом коде много вызовов функций, не являющихся членами классов, данное значение может быть уменьшено. Значение по умолчанию – 0.
TransformSwitchProbability	Вероятность преобразования switch в набор if.	Рекомендуемое значение – 100. если в обфусцируемом коде много операторов switch, данное значение может быть уменьшено.
TransformEntireSwitchProbability	Вероятность того, что при преобразовании switch в набор if это преобразование будет выполнено целиком.	Рекомендуемое значение – 100. если в обфусцируемом коде много операторов switch, содержащих много case, данное значение может быть уменьшено.
EncryptStringProbability	Вероятность того, что параметр функции, являющийся строковым литералом, будет заменён на передачу адреса массива, который будет инициализироваться перед вызовом функции.	Рекомендуемое значение – 0 или 100 в зависимости от того, нужно ли скрывать строковые литералы. При использовании этой возможности следует убедиться, что вызываемая функция не сохраняет адрес переданной строки для последующего использования, а копирует строку или использует её только во время работы функции.
EncryptArrayProbability	Вероятность того, что одномерные массивы целых чисел, инициализируемые константами, будут инициализироваться во время исполнения поэлементно.	Рекомендуемое значение – 0 или 100 в зависимости от того, нужно ли скрывать начальные значения в массивах.
EncryptVariableProbability	Вероятность того, что локальная переменная будет храниться в зашифрованном виде и расшифровываться при обращении.	Рекомендуемое значение – 10. Слишком большое значение может существенно увеличить размер кода и снизить скорость работы обфусцированной функции.
EncryptVariableReuseKeyProbability	Вероятность повторного использования ключа при шифровании переменной. Это нужно с целью использования одних и тех же ключей для разных переменных.	Рекомендуемое значение – 20. Не рекомендуется изменять указанное значение.
EncryptVariableUseXorProbability	Вероятность шифрования переменной с использованием операции xor (исключающее или).	Рекомендуемое значение – 50. Не рекомендуется изменять указанное значение.
StateChangeProbability	Вероятность генерации инструкции изменения состояния в базовом блоке.	Рекомендуемое значение – 80 в случае 8 переменных состояния, 40 для 4 переменных состояния, 90, если переменных состояния больше, чем 8. Не рекомендуется установка значения ниже 40 или выше 90. Уменьшение значения уменьшит размер кода, но может уменьшить уровень обфускации.
AdditionalStateChangeProbability	Вероятность генерации дополнительных инструкций изменения состояний в базовом блоке.	Рекомендуемый диапазон значений: 10-20. Значения выше 20 рекомендуются только если количество переменных состояний больше, чем 8. Увеличение данного значения увеличивает размер кода, а слишком большое значение может ухудшить уровень обфускации.
AdditionalStateChangeMaximumCount	Максимальное количество дополнительных инструкций изменения состояний в базовом блоке.	Рекомендуемое значение – 3.
MinimumBasicBlockSizeOnSplit-FirstStage	Минимальное количество инструкций в базовом блоке при разбиении блоков на первом этапе.	Рекомендуемое значение – 4. Не рекомендуется задавать значения ниже 3 или выше 6. Слишком маленькое значение

Параметр	Описание	Рекомендации по использованию
		существенно снизит скорость обфускации, но может ухудшить её уровень, несмотря на то, что размер результирующего кода будет большой. Слишком большое значение может ухудшить уровень обфускации, но размер кода будет меньше.
MaximumBasicBlockSizeOnSplit-FirstStage	Максимальное количество инструкций в базовом блоке при разбиении блоков на первом этапе.	Рекомендуемое значение – 5. Рекомендуется, чтобы данное значение было на 1 или 2 выше, чем предыдущее. Это нужно для того, чтобы базовые блоки были разного размера.
MinimumBasicBlockSizeOnSplitSecondStage	Минимальное количество инструкций в базовом блоке при разбиении блоков на втором этапе	Рекомендуемое значение – 3. Не рекомендуется задавать значения ниже 3 или выше 6. Слишком маленькое значение существенно снизит скорость обфускации, но может ухудшить её уровень, несмотря на то, что размер результирующего кода будет большой. Слишком большое значение может ухудшить уровень обфускации, но размер кода будет меньше.
MaximumBasicBlockSizeOnSplit SecondStage	Максимальное количество инструкций в базовом блоке при разбиении блоков на этапе.	Рекомендуемое значение – 4. Рекомендуется, чтобы данное значение было на 1 или 2 выше, чем предыдущее. Это нужно для того, чтобы базовые блоки были разного размера.
ReuseBlocksProbability	Вероятность слияния пар блоков, содержащих одинаковые инструкции, в один блок.	Рекомендуемое значение – 40. если значение MinimumPatternSizeForReuseBlocks равно 2, то может потребоваться уменьшение этой вероятности.
MinimumPatternSizeForReuseBlocks	Минимальный набор последовательностей инструкций для слияния блоков.	Рекомендуемое значение – 3. Увеличение снизит число доступных пар блоков. Не рекомендуется задавать значения большие, чем 3.
MaximumPatternSizeForReuseBlocks	Максимальный набор последовательностей инструкций для слияния блоков.	Рекомендуемое значение – 6. Если задать значение меньше 6, то скорость выполнения этапа слияния блоков может увеличиться.
DuplicateBasicBlockProbability	Вероятность дублирования базовых блоков.	Рекомендуемое значение – 20. Если задать слишком большое значение, то существенно увеличится объём обфусцируемого кода, вплоть до удвоения.
DuplicateLinkedBasicBlockProbability	Вероятность дублирования поддерева, то есть вероятность того, что после дублирования базового блока будет продолжено дублирование тех базовых блоков, на которые он ссылается.	Рекомендуемое значение – 60. Дублирование ограничивается также опцией MaximumNumberOfLinkedBasicBlocksDuplication, которая описана ниже. Чем выше её значение, тем меньше должно быть значение данной опции, с целью предотвращения дублирования больших частей кода.
MaximumNumberOfLinkedBasicBlocks-Duplication	Максимальное количество базовых блоков в поддереве.	Рекомендуемое значение – 5.
RelinkLinksToDuplicatedBlock-Probability	Вероятность перелинковки входных ссылок на дублированный блок.	Рекомендуемое значение – 50. Не рекомендуется изменять указанное значение.
DuplicateBasicBlockUsingUndefin	Вероятность дублирования с использованием неизвестной переменной	Рекомендуемое значение – 50. Не рекомендуется изменять указанное

Параметр	Описание	Рекомендации по использованию
ed-StateVariableFactor	состояния.	значение.
GenerateAssertProbability	Вероятность генерации ложных связей.	Рекомендуемое значение – 20. Если задать слишком большое значение, то существенно увеличится объём кода.
GenerateAssertUsingIdentityFactor	Вероятность генерации с использованием тождеств.	Рекомендуемое значение – 50. Не рекомендуется изменять указанное значение.
PartiallyResolveAssertProbability	Вероятность разрешения перехода на основном этапе.	Рекомендуемое значение – 10. Если задать слишком большое значение, то может ухудшиться уровень обфускации.
IntermixBlocksProbability	Вероятность вставки используемого базового блока в связь между другими двумя блоками.	Рекомендуемое значение – 40.
GenerateFictiveVariableAccess-Probability	Вероятность фиктивного доступа в неиспользуемые переменные.	Рекомендуемое значение – 40.
FictiveVariableAccessOnTransformingProbability	Вероятность фиктивного доступа в неиспользуемые переменные во время преобразования инструкций (TransformAssignmentProbability и TransformArithmeticProbability ниже).	Рекомендуемое значение – 60.
TransformAssignmentProbability	Вероятность преобразования инструкций присваивания.	Рекомендуемое значение – 15. Слишком большое значение может существенно увеличить размер кода и снизить скорость работы обфусцированной функции.
TransformArithmeticProbability	Вероятность преобразования инструкций сложения и вычитания.	Рекомендуемое значение – 15. Слишком большое значение может существенно увеличить размер кода и снизить скорость работы обфусцированной функции.
TransformLoadConstantProbability-OnSecondStage	Вероятность вычисления целочисленных констант во время исполнения.	Рекомендуемое значение – 15.
TransformLoadConstantProbabilityOnFinalStage	Вероятность вычисления целочисленных констант во время исполнения.	Рекомендуемое значение – 15.
TransformLoadConstantHighPriority-ProbabilityOnOnSecondStage	Вероятность вычисления целочисленных констант во время исполнения.	Рекомендуемое значение – 25.
TransformLoadConstant-HighPriority-ProbabilityOnOnFinalStage	Вероятность вычисления целочисленных констант во время исполнения.	Рекомендуемое значение – 25.
MergeEdgesProbabilityOnSecondStage	Вероятность объединения связей через специально введённый для этого блок.	Рекомендуемое значение – 15.
MergeEdgesProbabilityOnFinalStage	Вероятность объединения связей через специально введённый для этого блок.	Рекомендуемое значение – 20.
GeneratingFictiveTransientVariable-ProbabilityOnSe	Вероятность создания специальной переменной при объединении связей, через которую будут передаваться значения реальных данных с целью	Рекомендуемое значение – 80.

Параметр	Описание	Рекомендации по использованию
condStage	создания фиктивной связи по данным.	
GeneratingFictiveTransientVariable-ProbabilityOnFinalStage	Вероятность создания специальной переменной при объединении связей, через которую будут передаваться значения реальных данных с целью создания фиктивной связи по данным.	Рекомендуемое значение – 80.
FictiveVariableAccessWhileMergeEdges-ProbabilityOnSecondStage	Вероятность фиктивного доступа к переменным в переходном блоке при объединении связей.	Рекомендуемое значение – 60.
FictiveVariableAccess-WhileMergeEdgesProbability-OnFinalStage	Вероятность фиктивного доступа к переменным в переходном блоке при объединении связей.	Рекомендуемое значение – 60.
GenerateDispatcher	Генерировать ли диспетчер для данного уровня обфускации.	Рекомендуется включать только для функций, для которых не важна скорость выполнения.
DispatcherConstCalculationProbability	Вероятность того, что константа, которую нужно добавить, чтобы вычислить адрес следующего блока, будет вычисляться косвенным образом.	Рекомендуемое значение – 20.
DispatcherSecondIndexProbability	Вероятность генерации второго индекса (индекс фиктивный).	Рекомендуемое значение – 80.
DispatcherAdditionalIndexProbability	Вероятность генерации дополнительного индекса (индекс фиктивный).	Рекомендуемое значение – 20.
DispatcherMaximumAdditionalIndicesCount	Максимальное количество дополнительных индексов.	Рекомендуемое значение – 2.
DispatcherFictiveCasesProbability	Вероятность генерации дополнительных фиктивных индексов при вычислении констант для передачи управления. Наличие таких индексов усложняет поиск блоков, на которые может осуществляться переход из данного блока.	Рекомендуемое значение – 60.
DispatcherMaximumFictiveCasesCount	Максимальное количество дополнительных индексов (см. описание DispatcherFictiveCasesProbability).	Рекомендуемое значение – 2.
OutlineProbability	Вероятность того, что базовый блок, который может быть вынесен в отдельную функцию, будет вынесен.	Рекомендуемое значение – 0 или 10. Рекомендуется задавать отличное от нуля значение только для функций, для которых не важна скорость выполнения.
OutlineFullyOutlineableGroups	Выносить все блоки из группы в отдельную функцию, если все блоки в группе могут быть вынесены (блоки, содержащие доступ к private/protected-членам класса, не могут быть вынесены).	Рекомендуется включать только для функций, для которых не важна скорость выполнения.
MergeBlocksProbability	Вероятность объединения блоков, имеющих простую связь, с целью уменьшения их количества.	Рекомендуемое значение – 100.

Настройки для каждого уровня обфускации

6 Миграция со старой версии StarForce C++ Obfuscator 1.X на новую версию 2.X

Новая версия StarForce C++ Obfuscator 2.X имеет отличия в применении от старой версии StarForce C++ Obfuscator 1.X. Они заключаются в следующем:

1. Изменён синтаксис атрибута для указания того, что функция должна быть обфусцирована.

   Поэтому при миграции замените __attribute__((obfuscate(<уровень_обфускации>))) на __attribute__((annotate("StarForce::Obfuscator::Obfuscate(<уровень_обфускации + 1>)"))).

2. Изменена нумерация уровней обфускации. В старой версии обфускатора по умолчанию уровни обфускации нумеровались от 0 (самая слабая обфускация) до 4 (самая сильная обфускация). В новой версии обфускатора уровень 0 соответствует отсутствию обфускации и используется при отладке, а в реальной обфускации используются уровни от 1 (самая слабая обфускация) до 5 (самая сильная обфускация).

   При миграции увеличьте значения всех уровней обфускации на 1.

3. Изменён синтаксис конфигурационного файла параметров компилятора (TargetConfig). Старый конфигурационный файл не будет работать с новым обфускатором.

   При миграции замените используемый TargetConfig на новый. Его можно выбрать из идущих в комплекте примеров, либо написать самостоятельно.

Кроме того, в старой версии обфускатора для совместимости со стандартной библиотекой C++ проект необходимо было перевести на использование альтернативной реализации стандартной библиотеки STLport. В новой версии обфускатора этого не требуется, так как он совместим со всеми основными реализациями STL.