Лекция 13

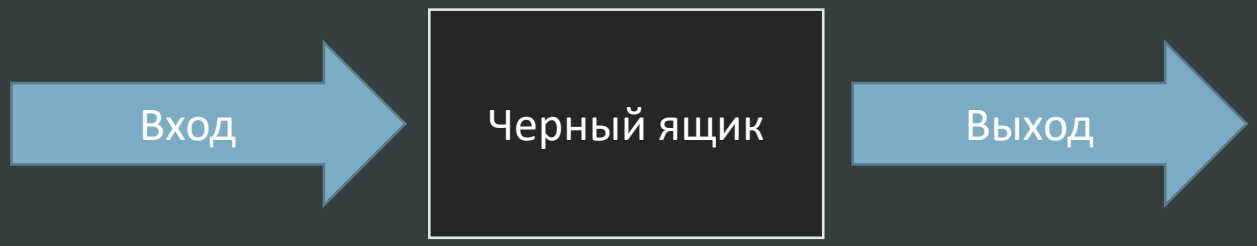
ВВЕДЕНИЕ В РЕВЕРС-ИНЖИНИРИНГ

rev. 5

Что такое реверс-инжиниринг

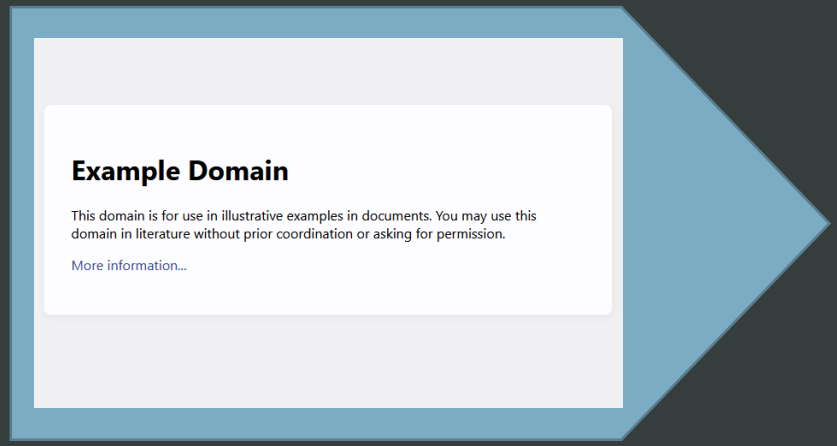
Концепция «черного ящика»

- Детали работы большинства программ и устройств, с которыми вы работаете, вам скорее всего неизвестны
- Такие системы называются «черными ящиками»: вы можете иметь представление об их функциях как пользователь, но не более того
- Черные ящики обычно представляют следующей схемой:



Пример черного ящика

http://example.com



Что из программ черным ящиком не является?

- Программа, которую вы сами написали*
- Программа, которую вы собрали из исходного кода, причем прочитали и поняли весь код* **
- Программа, которая побайтово (за исключением данных о времени сборки и т.д.) эквивалентна той, что вы можете собрать* из исходного кода (который вы, естественно, прочитали и поняли **)
- Chrome с предыдущего слайда, в отличие от Chromium, совсем не подходит

* при условии, что исходный код компилятора вы тоже прочитали и поняли

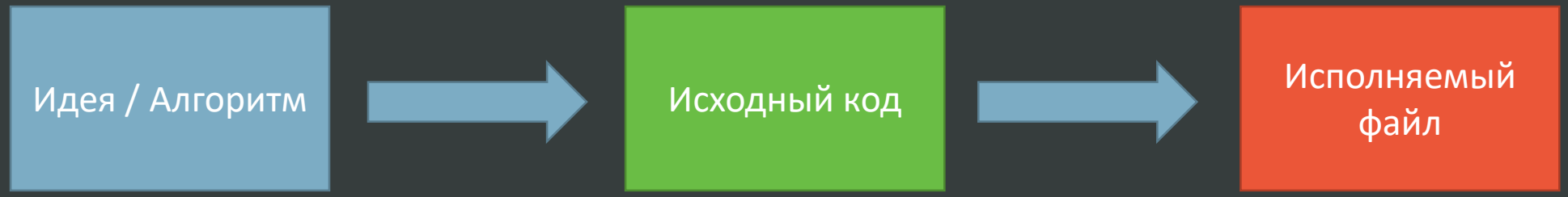
** при условии, что программа не содержит удаленных механизмов управления

Реверс-инжиниринг

- Также переводится как «обратная разработка», иногда сокращенно называется «реверс»
- Процесс исследования некоторой готовой программы или устройства с целью понимания принципа его работы
  - Иначе говоря, реверс делает из черного ящика – белый
- Может нарушать авторское право и патентное законодательство

Реверс-инжиниринг

Нормальная разработка:

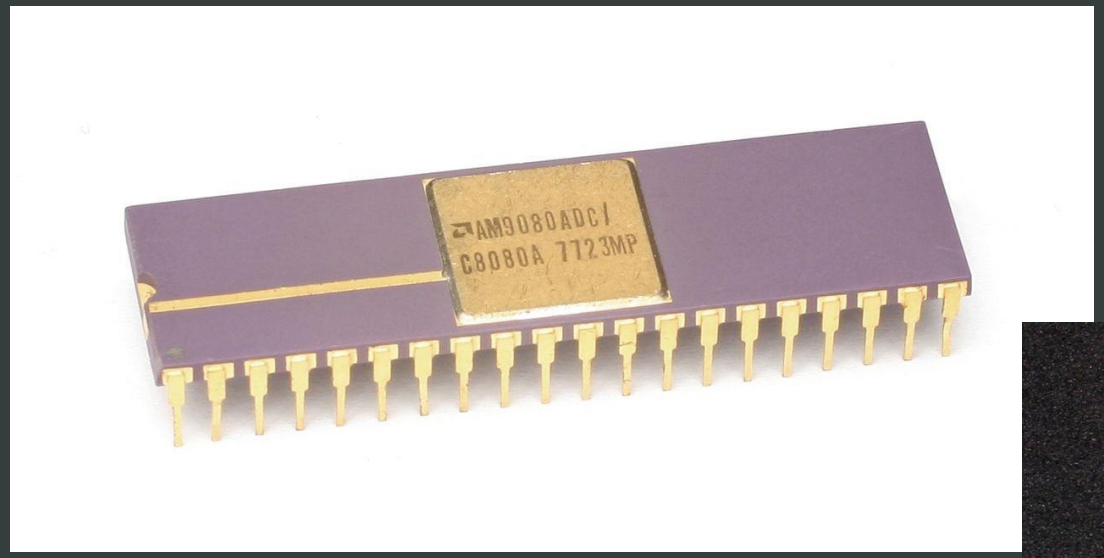


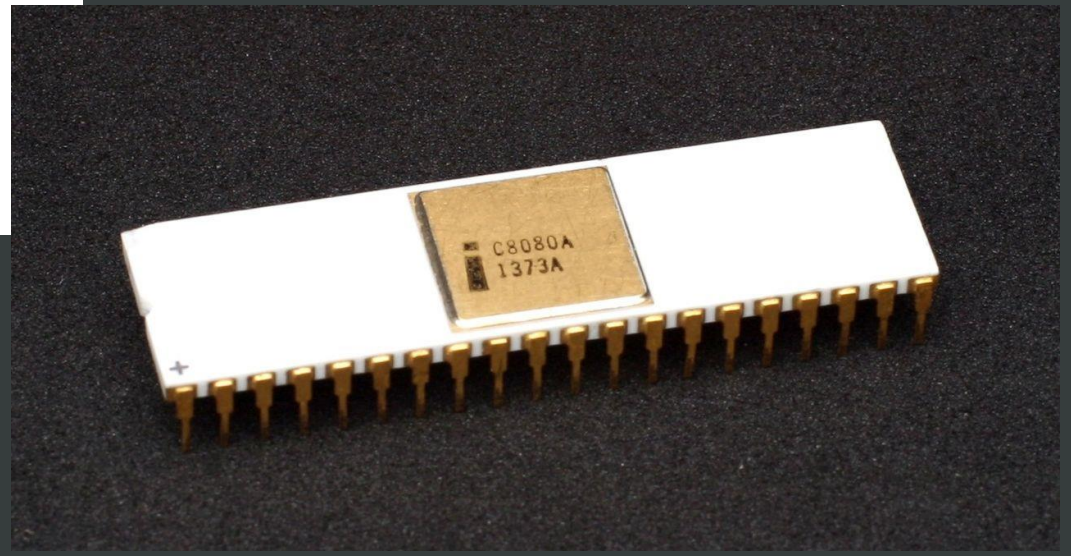
Обратная разработка:

Примеры обратной разработки

- AMD Am9080 – безлицензионный клон Intel 8080
  - Разработан путем изучения оригинального Intel 8080 под микроскопом
- Электроника-60 – клон PDP-11
  - Как, впрочем, и многая другая советская электроника тех времен
- Китайская военная техника
  - Без конкретных примеров, так как их слишком много

Примеры обратной разработки





Примеры обратной разработки ПО

- Wine – реализация Windows API для Linux
  - ReactOS – копия Windows, только с совместимостью на уровне ядра, использует Wine
  - Proton – успешный форк Wine от Valve, используемый в игровой консоли Steam Deck
- Samba – реализация протокола SMB для Linux
  - Сетевые папки в Windows работают как раз по этому протоколу
- Большинство кейгенов, «кряков» и активаторов разработаны с использованием обратной разработки
- Некоторые из Open Source ремейков старых игр:
  - Xash3D – реализация движка Half Life
  - OpenMW – реализация движка TES III: Morrowind
  - RE3 – реализация движка GTA III / Vice City

Цели обратной разработки ПО

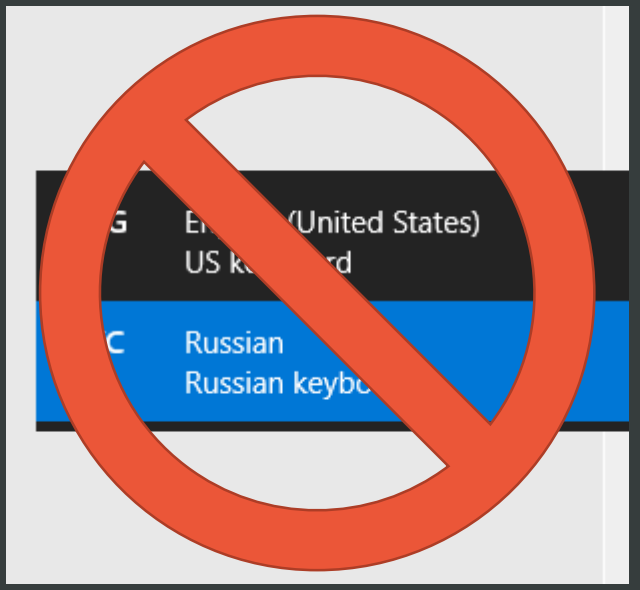
- Исследование чужих программ с целью копирования
  - Отличный способ сэкономить на разработке и исследованиях, а также увеличить скорость разработки
- Исследование чужих программ с целью интеграции
  - В случае, если какая-то программа или сервис не дают вам нормальный API для работы, а вам очень хочется
  - Пример: неофициальные API для WhatsApp
- Исследование чужих программ с целью взлома
  - В случае, если вам не хочется платить за программы (ну или вы за них заплатили, а потом правообладатель решил их отобрать, что часто встречалось в 2022 году)
  - Пример: всевозможные crack и keygen, доступные в интернете

Цели обратной разработки ПО

- Восстановление потерянных исходников
  - Или внесение каких-то срочных изменений в готовую программу
- Анализ вредоносного ПО
  - Исходный код вредоносного ПО как правило недоступен, а понять, как оно работает, нужно
- Исследование чужих программ с целью поиска уязвимостей
  - Об этом мы поговорим во второй теме семестра
  - Примерами являются практически все уязвимости, найденные в Windows, Office и т.д.

Цели обратной разработки ПО

- Исправление чужих программ
  - Удаление из Windows сомнительного окна переключения языков (причем, я не единственный, кто исправлял это дизассемблером: https://answers.microsoft.com/en-us/windows/forum/all/how-to-turn-off-language-switch-bar/a9ed3821-af73-4a06-a105-f725d6e5fdf0)
  - Улучшение интеграции драйверов стриминга NVIDIA для использования в качестве удаленного рабочего стола с Moonlight Streaming (отключение принудительного включения «поверх всех окон» и т.д.)
  - Исправление вылетов в классических играх, например NFS: MW: https://habr.com/ru/post/349296/
- В целях развлечения и участия в соревнованиях CTF



Законность реверс-инжиниринга

- Реверс инжиниринг не особо законен, и законодательство различных стран, включая РФ, зачастую его запрещает

Есть следующие способы осуществлять его без риска преследования:

- Использовать реверс-инжиниринг для починки багов и добавления возможности взаимодействия со своими программами, как правило это разрешено (ГК 1280)
- Реверсить вредоносное ПО: вряд ли его автор заявит на вас в полицию
- Никому не рассказывать о том, что вы им занимаетесь. Если реверс-инжиниринг осуществляется на ПК, отрезанном от интернета, вряд ли кто-то что-то узнает
- Использовать принцип «Clean Room» (как, например, был создан Wine): один человек пишет ТЗ, другой пишет по нему код. Опционально между ними может быть юрист, проверяющий что в ТЗ нет ничего лишнего. От нарушений патентов это, естественно, не спасет, но копирования кода точно получится избежать

Виды реверс-инжиниринга

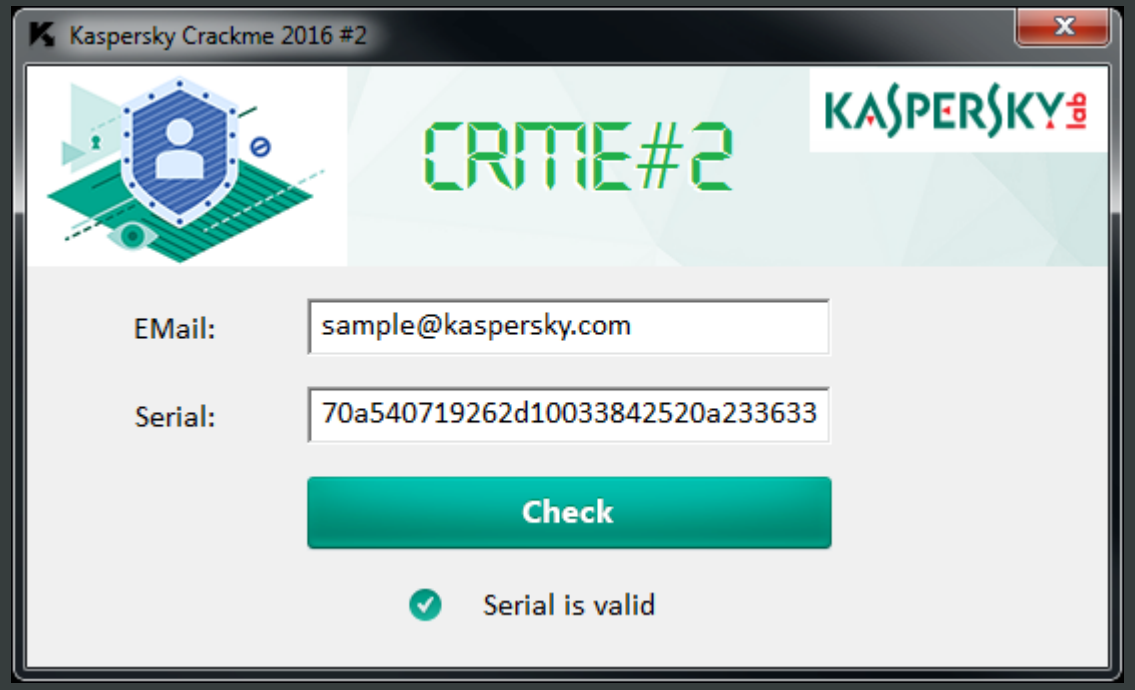
1. Реверс-инжиниринг программ с исходным кодом
   - Звучит немного странно, но тем не менее иногда актуально
2. Реверс-инжиниринг программ в виде машинного кода
   - Сюда относятся большинство программ, которые вы запускаете на своем компьютере
3. Реверс-инжиниринг программ, использующих байт-код
   - Сюда относятся программы на Java, .NET, Python и т.д.

Реверс как развлечение, CrackMe

CrackMe

- Программы CrackMe стали популярны еще до CTF
  - Они распространялись на форумах и специализированных сайтах
  - Также при помощи CrackMe антивирусные компании любят нанимать себе ~~рабов~~ сотрудников
- Целью решения CrackMe является поиск какого-либо пароля, который проверяет программа CrackMe или написание кейгена
- Как правило, CrackMe это исполняемые файлы для Windows
  - В отличие от заданий CTF, где Windows популярностью не пользуется

CrackMe


Kaspersky Crackme 2016 #2
CRME#2
KASPERSKY lab
EMail:
sample@kaspersky.com
Serial:
70a540719262d10033842520a233633
Check
Serial is valid

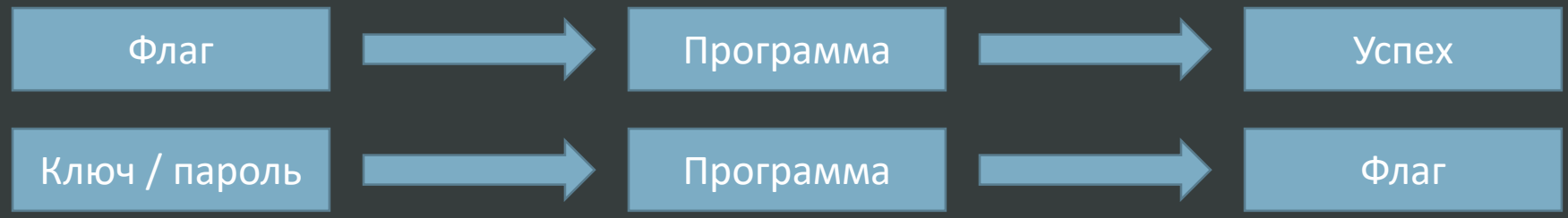
CrackMe. Пример исходного кода

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
        char string[1024];
        printf("Hello, please enter key: ");
        scanf("%1023[^\n]", string);
        if (!strcmp(string, "someflag"))
                printf("Correct key, congratulations!\n");
        else
                printf("Wrong key\n");
        return 0;
}
```

CrackMe и этот курс

- Большинство программ этого раздела будут в том или ином виде являться CrackMe
- В CTF также существуют задания на реверс, где надо, например, «исправить баг в программе»
  - Конкретно этот вид реверса мне не слишком нравится, да и образовательной пользы в нем немного
- Основные виды заданий:



Время задач

Simplest CrackMe

Категория: Lesson 13 / Reverse basics

Решивших: 0

Время: 00:00:02

- Доступ к задачам можно получить как обычно на nsuctf.ru

Реверс программ с доступным исходным кодом

Реверс с исходным кодом

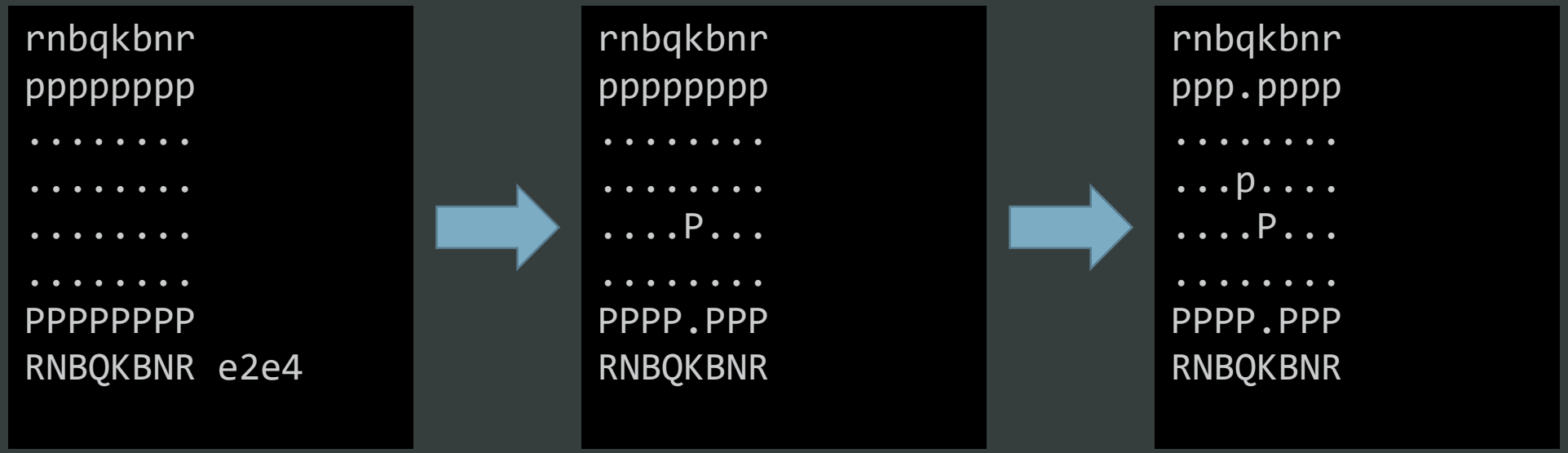
- Как вы помните, для того, чтобы черный ящик стал белым, нужно не только, чтобы код был у вас на руках, но и чтобы вы его поняли
  - И с этим пунктом бывают проблемы
- Написание самых вопиющих примеров непонятных программ на языке Си является международным соревнованием International Obfuscated C Code Contest
  - Там есть круглая программа для подсчета числа π
  - И авиасимулятор в виде самолета
- Также существует такое понятие как «эзотерические языки программирования»
  - О них чуть позже

Реверс с исходным кодом

```
char*l="ustvrtsuqqqqqqqqyyyyyyyy}{|~z|{}"
"   76Lsabcddcba .pknbrq  PKNBRQ ?A6J57IKJT576,+-48HLSU";
#define F getchar()&z
#define v X(0,0,0,21,
#define Z while(
#define _ ;if(
#define P return--G,y^=8,
B,i,y,u,b,I[411],*G=I,x=10,z=15,M=1e4;X(w,c,h,e,S,s){int t,o,L,E,d,O=e,N=-M*M,K
=78-h<<x,p,*g,n,*m,A,q,r,C,J,a=y?-x:x;y^=8;G++;d=w||s&&s>=h&&v 0,0)>M;do{_ o=I[
p=O]){q=o&z^y _ q<7){A=q--&2?8:4;C=o-9&z?q["& .$  "]:42;do{r=I[p+=C[l]-64]_!w|p
==w){g=q|p+a-S?0:I+S _!r&(q|A<3||g)||(r+1&z^y)>9&&q|A>2){_ m=!(r-2&7))P G[1]=O,
K;J=n=o&z;E=I[p-a]&z;t=q|E-7?n:(n+=2,6^y);Z n<=t){L=r?l[r&7]*9-189-h-q:0 _ s)L
+=(1-q?l[p/x+5]-l[O/x+5]+l[p%x+6]*-~!q-l[O%x+6]+o/16*8:!!m*9)+(q?0:!(I[p-1]^n)+
!(I[p+1]^n)+l[n&7]*9-386+!!g*99+(A<2))+!(E^y^9)_ s>h||1<s&s==h&&L>z|d){p[I]=n,O
[I]=m?*g=*m,*m=0:g?*g=0:0;L-=X(s>h|d?0:p,L-N,h+1,G[1],J=q|A>1?0:p,s)_!(h||s-1|B
-O|i-n|p-b|L<-M))P y^=8,u=J;J=q-1|A<7||m||!s|d|r|o<z||v 0,0)>M;O[I]=o;p[I]=r;m?
*m=*g,*g=0:g?*g=9^y:0;}_ L>N){*G=O _ s>1){_ h&&c-L<0)P L _!h)i=n,B=O,b=p;}N=L;}
n+=J||(g=I+p,m=p<O?g-3:g+2,*m<z|m[O-p]||I[p+=p-O]);}}}}Z!r&q>2||(p=O,q|A>2|o>z&
!r&&++C*--A));}}}Z++O>98?O=20:e-O);P N+M*M&&N>-K+1924|d?N:0;}main(){Z++B<121)*G
++=B/x%x<2|B%x<2?7:B/x&4?0:*l++&31;Z B=19){Z B++<99)putchar(B%x?l[B[I]|16]:x)_
x-(B=F)){i=I[B+=(x-F)*x]&z;b=F;b+=(x-F)*x;Z x-(*G=F))i=*G^8^y;}else v u,5);v u,
1);}}
```

Реверс с исходным кодом

- Как вы наверное догадались, это шахматная программа
- У нее даже искусственный интеллект есть



Обфускация

- Процесс приведения исходного кода к виду, в котором он сохраняет функциональность, однако выглядит совсем не читаемо
  - Существует также обфускация на уровне машинного кода, но мы сейчас не будем ее рассматривать
  - Особенно популярна обфускация при применении к некомпилируемым языкам
- Инструменты, осуществляющие обфускацию, называются обфускаторами

```
#include <stdio.h>
int main() {
        printf("Hello World!\n");
}
```

```
#include <stdio.h>
int main(){printf("\x48""e\154l\x6F"" \127o\x72""l\144!\n");};
```

Обфускация

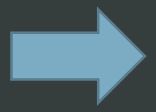
```
/*!
 * jQuery JavaScript Library v3.5.1
 * https://jquery.com/
 *
 * Includes Sizzle.js
 * https://sizzlejs.com/
 *
 * Copyright JS Foundation and other contributors
 * Released under the MIT license
 * https://jquery.org/license
 *
 * Date: 2020-05-04T22:49Z
 */
( function( global, factory ) {

    "use strict";

    if ( typeof module === "object" && typeof module.exports === "object" ) {

        // For CommonJS and CommonJS-like environments where a proper `window`
        // is present, execute the factory and get jQuery.
        // For environments that do not have a `window` with a `document`
        // (such as Node.js), expose a factory as module.exports.
        // This accentuates the need for the creation of a real `window`.
        // e.g. var jQuery = require("jquery")(window);
        // See ticket #14549 for more info.
        module.exports = global.document ?
            factory( global, true ) :
            function( w ) {
                if ( !w.document ) {
                    throw new Error( "jQuery requires a window with a document" );
                }
                return factory( w );
            };
    } else {
        factory( global );
    }

// Pass this if window is not defined yet
} )( typeof window !== "undefined" ? window : this, function( window, noGlobal ) {
```



```
/*! jQuery v3.5.1 | (c) JS Foundation and other contributors | jquery.org/license */
!function(e,t){"use strict";"object"==typeof module&&"object"==typeof
module.exports?module.exports=e.document?t(e,!0):function(e){if(!e.document)throw new Error("jQuery
requires a window with a document");return t(e)}:t(e)}("undefined"!=typeof
window?window:this,function(C,e){"use strict";var
t=[],r=Object.getPrototypeOf,s=t.slice,g=t.flat?function(e){return t.flat.call(e)}:function(e){return
t.concat.apply([],e)},u=t.push,i=t.indexOf,n={},o=n.toString,v=n.hasOwnProperty,a=v.toString,l=a.call(
Object),y={},m=function(e){return"function"==typeof e&&"number"!=typeof
e.nodeType},x=function(e){return
null!=e&&e===e.window},E=C.document,c={type:!0,src:!0,nonce:!0,noModule:!0};function b(e,t,n){var
r,i,o=(n=n||E).createElement("script");if(o.text=e,t)for(r in
c)(i=t[r]||t.getAttribute&&t.getAttribute(r))&&o.setAttribute(r,i);n.head.appendChild(o).parentNode.re
moveChild(o)}function w(e){return null==e?e+"":"object"==typeof e||"function"==typeof
e?n[o.call(e)]||"object":typeof e}var f="3.5.1",S=function(e,t){return new S.fn.init(e,t)};function
p(e){var t=!!e&&"length"in
e&&e.length,n=w(e);return!m(e)&&!x(e)&&("array"===n||0===t||"number"==typeof t&&0<t&&t-1 in
e)}S.fn=S.prototype={jquery:f,constructor:S,length:0,toArray:function(){return
s.call(this)},get:function(e){return
null==e?s.call(this):e<0?this[e+this.length]:this[e]},pushStack:function(e){var
t=S.merge(this.constructor(),e);return t.prevObject=this,t},each:function(e){return
S.each(this,e)},map:function(n){return this.pushStack(S.map(this,function(e,t){return
n.call(e,t,e)}))},slice:function(){return
this.pushStack(s.apply(this,arguments))},first:function(){return this.eq(0)},last:function(){return
this.eq(-1)},even:function(){return
this.pushStack(S.grep(this,function(e,t){return(t+1)%2}))},odd:function(){return
this.pushStack(S.grep(this,function(e,t){return t%2}))},eq:function(e){var
t=this.length,n=+e+(e<0?t:0);return this.pushStack(0<=n&&n<t?[this[n]]:[])},end:function(){return
this.prevObject||this.constructor()},push:u,sort:t.sort,splice:t.splice},S.extend=S.fn.extend=function
(){var e,t,n,r,i,o,a=arguments[0]||{},s=1,u=arguments.length,l=!1;for("boolean"==typeof
a&&(l=a,a=arguments[s]||{},s++),"object"==typeof a||m(a)||(a={}),s===u&&(a=this,s--
);s<u;s++)if(null!=(e=arguments[s]))for(t in
e)r=e[t],"__proto__"!==t&&a!==r&&(l&&r&&(S.isPlainObject(r)||(i=Array.isArray(r)))?(n=a[t],o=i&&!Array
.isArray(n)?[]:i||S.isPlainObject(n)?n:{},i=!1,a[t]=S.extend(l,o,r)):void 0!==r&&(a[t]=r));return
a},S.extend({expando:"jQuery"+(f+Math.random()).replace(/\D/g,""),isReady:!0,error:function(e){throw
new Error(e)},noop:function(){},isPlainObject:function(e){var t,n;return!(!e||"[object
Object]"!==o.call(e))&&(!(t=r(e))||"function"==typeof(n=v.call(t,"constructor")&&t.constructor)&&a.cal
l(n)===l)},isEmptyObject:function(e){var t;for(t in
e)return!1;return!0},globalEval:function(e,t,n){b(e,{nonce:t&&t.nonce},n)},each:function(e,t){var
n,r=0;if(p(e)){for(n=e.length;r<n;r++)if(!1===t.call(e[r],r,e[r]))break}else for(r in
e)if(!1===t.call(e[r],r,e[r]))break;return e},makeArray:function(e,t){var n=t||[];retur
```

Эзотерические языки программирования

- Вид языков программирования, созданных со следующими целями:
  - Для проверки какой-либо концепции (например, создание языка с самым маленьким компилятором)
  - В качестве шутки или головоломки
- Использование таких языков программирования можно также считать формой запутывания кода
  - Например, интерпретатор такого языка может быть встроен в проверку лицензии программы
- Одним из самых известных представителей является Brainfuck

Brainfuck

- Язык программирования, имеющий следующие команды:
  - >, < – переход к следующей и предыдущей ячейкам данных
  - +, - – увеличение и уменьшение значения в текущей ячейке данных
  - . – вывод содержимого текущей ячейки данных на экран
  - , – ввод символа с клавиатуры в текущую ячейку данных
  - [,] – операторы циклов
    - [переходит на фрагмент текста программы сразу за соответствующим], если значение текущей ячейки ноль
    -] переходит к команде сразу за соответствующим [, если значение текущей ячейки не ноль
- Является полным по Тьюрингу при бесконечном числе ячеек

Brainfuck

Команда Brainfuck	Замена на Си
>	i++;
<	i--;
+	arr[i]++;
-	arr[i]--;
.	putchar(arr[i]);
,	arr[i] = getchar();
[	while(arr[i]){
]	}

Brainfuck

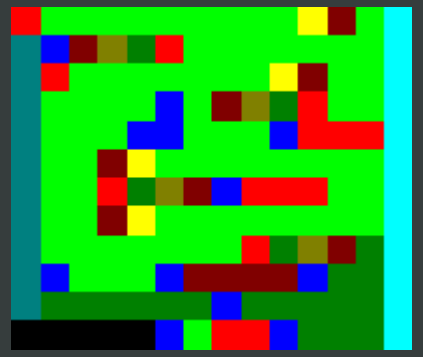
- Пример программы, выводящей "Hello World!" выглядит следующим образом:

```
++++++++[>++++[>++>+++>+++>+<<<<-]>+>+>->>+[<]<-]>>.>---
.+++++++..+++.>>.<-.<.+++.------.--------.>>+.>++.
```

- Как можно видеть, язык не очень понятный, хотя и достаточно простой
- У Brainfuck множество последователей среди эзотерических языков, тривиально приводимых к нему путем замены символов / слов:
  - Ook!
  - Blub
  - (Вероятно) полный список можно посмотреть на https://esolangs.org/wiki/Joke_language_list#Brainfuck_derivatives

Обнаруживаем Brainfuck

- Самым простым способом обнаружить Brainfuck является подсчет числа используемых команд
  - Если их 8 (или меньше), то высока вероятность, что вы имеете дело с Brainfuck
- Также из-за своей специфики Brainfuck может требовать повторения одной и той же команды несколько раз подряд (+ и -), что может быть заметно в коде
- Подобная логика обнаружения может быть с некоторым успехом применена и к необычным вариантам Brainfuck, например Brainloller:
  - Можно заметить, что за исключением «стен» по бокам программы, в остальных ее местах используется всего 7 команд



Прочие непонятные языки

- Есть и другие языки, которые некоторые программисты вполне могли бы отнести к эзотерическим:
- Языки с парадигмами, отличными от императивного программирования:
  - Самый классический пример – Lisp
- Не очень популярные языки, сильно отличные от Си-подобных:
  - 1C, Pascal
  - Basic, вопреки своему названию
- Perl

Общие принципы реверса

Общие принципы реверса

- Среди полезных трюков, не зависящих от того, что вы реверсите, я бы отметил следующие:
  - Удаление неудобного кода
  - Наблюдение за сравнениями
  - Поиск популярных констант
  - Сборка-декомпиляция

Удаление неудобного кода

- Иногда в CTF-заданиях бывает такое, что программа требует пароль (не флаг) и показывает флаг, только если он верен
  - Однако при генерации флага пароль не используется
- В этом случае можно просто удалить неудобную функцию проверки
- В реальных программах Crack (в отличие от Keygen) обычно работает именно так
  - По причине простоты подхода он обычно менее уважаем, чем создание кейгенов

```
...
int main()
{
    char password[1024];
    gets(password);
    if (check_password(password))
        print_flag();
}
```

```
...
int main()
{
    char password[1024];
    gets(password);
    print_flag();
}
```

Удаление неудобного кода. Расширенный пример

- Здесь принцип тот же, хотя для генерации флага и используется и значение, являющееся функцией от пароля, верное значение известно
- В роли сложной функции может выступать криптографическая хэш-функция
- Кстати, если вы встретили криптографическую хэш-функцию на CTF-соревнованиях, нужно иметь в виду, что она вполне может быть необратима перебором
  - И вообще перебор хэшей – не самое достойное для CTF занятие

```
...
int main()
{
    char password[1024];
    gets(password);
    uint32_t fval = hard_function(password);
    if (fval == 0x1337)
        print_flag(fval);
}
```

```
...
int main()
{
    print_flag(0x1337);
}
```

Наблюдение за сравнениями

- Иногда бывает такое, что задание сравнивает сгенерированный флаг с тем, что мы ввели при помощи strcmp (или чего-то в этом духе)
- В этом случае можно просто посмотреть, что передается в strcmp и взять оттуда флаг
  - Существуют способы сделать это и без обладания исходным кодом, но об этом в следующих лекциях
- В реальном мире это отличное место для создания Keygen

```
...
int main()
{
    char key[1024];
    gets(key);
    if (!strcmp(key, gen_key()))
        puts("Ай молодца");
}
```

```
...
int main()
{
    puts(gen_key());
}
```

Поиск популярных констант

- Многие математические и особенно криптографические алгоритмы содержат в своей реализации некоторые магические значения констант
- Не обязательно читать код алгоритма, чтобы понять, что за алгоритм используется
- Только лучше перепроверить, что это действительно он, вдруг кто-то решил вас перехитрить
  - Например, я так делал в некоторых из задач, где нужно было сделать код неприятным для реверса «в лоб»

```
...
char* unknown_function(char* data)
{
    uint32_t h0 = 0x67452301;
    uint32_t h1 = 0xefcdab89;
    uint32_t h2 = 0x98badcfe;
    uint32_t h3 = 0x10325476;
    ...
```

stackoverflow.com › questions ▾ Перевести эту страницу

question on MD5 state variables - Stack Overflow

1 ответ

9 нояб. 2011 г. — Those variables are **0x67452301** , 0xEFCDAB89, 0x98
0x10325476. I converted variables to decimals and came up with 173258

```
...
char* md5_hash(char* data)
{
    ...
```

Время задач

Сложный алгоритм

Категория: Lesson 13 / Reverse basics

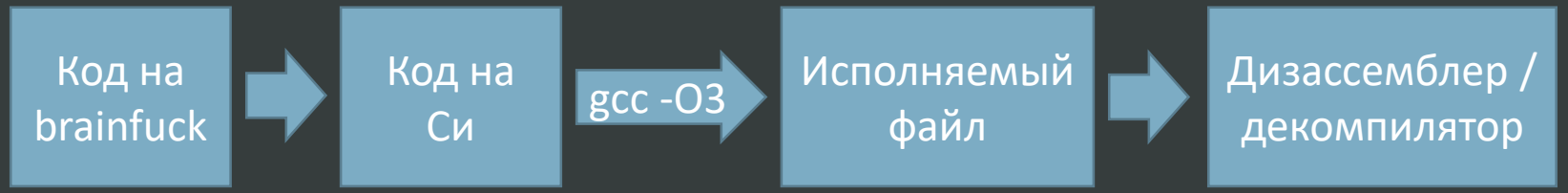
Решивших: 0

Время: 00:00:02

- Доступ к задачам можно получить как обычно на nsuctf.ru

Сборка-декомпиляция

- Иногда исходный код настолько плох, что в виде машинного кода он лучше
  - Особенно хорошо это работает с эзотерическими языками программирования
- В этом случае, можно собрать исполняемый файл, а потом разобрать его, используя способы применимые к чистому машинному коду
- Самый классный вариант для Brainfuck-подобных языков:




```
++++[++++>---<]>++.-[-----
>+<]>++++.--[->+++<]>-
.++++++++++++.+++.----.-------.
```

```
; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near
; __unwind {
sub     rsp, 8
mov     rsi, cs:__bss_start ; fp
mov     edi, 45h ; 'E'  ; c
call    __IO_putc
mov     rsi, cs:__bss_start ; fp
mov     edi, 78h ; 'x'  ; c
call    __IO_putc
mov     rsi, cs:__bss_start ; fp
mov     edi, 61h ; 'a'  ; c
call    __IO_putc
mov     rsi, cs:__bss_start ; fp
mov     edi, 6Dh ; 'm'  ; c
call    __IO_putc
mov     rsi, cs:__bss_start ; fp
mov     edi, 70h ; 'p'  ; c
call    __IO_putc
mov     rsi, cs:__bss_start ; fp
mov     edi, 6Ch ; 'l'  ; c
call    __IO_putc
mov     rsi, cs:__bss_start ; fp
mov     edi, 65h ; 'e'  ; c
call    __IO_putc
xor     eax, eax
add     rsp, 8
retn
; } // starts at 580
main endp
```

Как защититься от реверса

- Проверяйте лицензионные ключи при помощи асимметричной криптографии
  - Тогда вы пройдете мимо большинства типичных граблей, т.к. задача из программистской станет математической
- Проверяйте целостность своих программ, это помешает использовать более простой подход создания Crack к вашей программе
  - А вот это сделать уже непросто, в дальнейшем мы рассмотрим, как могут обходиться различные проверки
- Вынесите часть программной логики на удаленный сервер
  - Легендарные примеры: Diablo III, Assassin's creed II
- Никогда не включайте в код программы то, секретность чего является критической
  - Ключи симметричного шифрования рано или поздно точно станут общим достоянием
  - Всякие API-ключи тоже желательно в клиенте не хранить (хотя и не всегда возможно)

Немного организационных вопросов

- Рекомендую всем обзавестись виртуальной машиной с Linux
  - В качестве дистрибутива хорошо подойдет Ubuntu LTS (на ваш выбор)
  - В качестве виртуальной машины хорошо подходит VMWare, но VirtualBox тоже сойдет
  - WSL по началу тоже сойдет, но потом его будет не хватать
- Реверс под Windows в основной части курса не изучается
  - Хотя общие принципы те же и там и там, но Linux проще для обучения
  - Также будет дополнительная лекция про значимые отличия реверса для Windows
- В качестве ОС хоста, однако, рекомендую Windows (если у вас почему-то не она)
  - Основные программы для реверса там запускаются комфортнее

Спасибо за внимание!
Задачи доступны на

nsuctf.ru

- Пожалуйста, используйте имя пользователя формата “Фамилия Имя”
  - e-mail можно забить любой, сервером он не проверяется
- Для вопросов по задачам рекомендую присоединиться к @NSUCTF в Telegram
  - Только, пожалуйста, без спойлеров