UCRP 1.30, описание формата выходных файлов и используемых алгоритмов.

В данном документе рассматривается формат архивных файлов, генерируемых программой ucrp версии 1.30. Более ранние форматы не рассматриваются. В приложении A к данному дкументу приведена лицензия программы ucrp 1.30, лицензия свободно распространяемой, усечённой версии ucrp 1.30. В приложении B рассматриваются опции командной строки.

Документ подготовлен утилитой docbook2pdf из состава пакета DocBook-utils version 0.6.14 (jw version 1.1), поставляемого в составе ASPLinux 7.3 (ядро vmlinuz-2.6.9-1.667asp), для просмотра документа использовался xpdf версии 3.0, также входящий в состав ASPLinux 7.3.

Table of Contents

Заголовок параметров, ключевой информации, блоки данных

Файл состоит из трех частей:

- **8192 байт, заголовок параметров, задаёт используемые алгоритмы и некоторые их параметры.**
- **8192 байт, ключевая информация, как правило псевдослучайные параметры, используемые алгоритмами обработки данных.**
- **блоки данных. Размер блока данных по умолчанию 65536 байт исходного файла. При включённом алгоритме ECC на каждые 256 байт полных или неполных добавляется 3 байта проверочных.**

1. Порядок записи битовых полей. Величины смещений битовых полей.

Старший/наиболее значимый разряд битового поля записывается в старший/наиболее значимый бит записываемого в файл байта данных. Номер байта в который попадает бит с номером <номер бита> задаётся как <номер бита>/8, маска бита с номером <номер бита>, записываемого байта задаётся как (1 << (<номер бита> & 7)), где '<<' операция битового сдвига числа, в данном случае 1, в сторону наиболее значимых разрядов, '/' - операция целочисленного деления, а '\& 7' операция извлечения/маскирования трех младших бит. Величины смещений, указанные в дальнейших пунктах отсчитываются от начала соответствующего заголовка.

2.Заголовок параметров.

2.1. Название программы, её версия.

8 байт со смещением 0. Строка типа ucrp1.30 для версии программы 1.30.

2.2. <бит xor> бит включения алгоритма <xor>.

Номер бита 64. Определяет обрабатываются входные данные алгоритмом <xor> или нет.

2.3. <бит add> бит включения алгоритма <add>.

Номер бита 65. Определяет обрабатываются входные данные алгоритмом <add> или нет.

2.4. <бит rot> бит включения алгоритма <rot>.

Номер бита 66. Определяет обрабатываются входные данные алгоритмом <rot>; или нет.

2.5. <бит bts> бит включения алгоритма <bts>.

Номер бита 67. Определяет обрабатываются входные данные алгоритмом <bts> или нет.

2.6. <бит tab> бит включения алгоритма <tab>.

Номер бита 68. Определяет обрабатываются входные данные алгоритмом <tab> или нет.

2.7. <бит ecc> бит включения алгоритма <ecc>.

Номер бита 69. Определяет обрабатываются входные данные алгоритмом <ecc> или нет.

2.8. Размер исходного буфера данных в байтах.

Номер стартового бита 72, длина поля 32 бита. Исходный, не кодированный файл читается блоками указанного размера. После применения к считанному блоку данных перечисленных ниже алгоритмов формируется выходной блок, который записывается в выходной файл. При нулевом значении данного поля используется значение по умолчанию 65536 байт.

2.9. Размер выходного блока данных

Номер стартового бита 104, длина поля 32 бита. При нулевом значении данного поля используется значение, которое расчитывается по используемому размеру исходного буфера данных.

3. Заголовок ключевой информации.

3.1. <маска xor>.

Смещение 0, длина битового поля 8.

3.2. <слагаемое>.

Смещение 8, длина битового поля 8.

3.3. <сдвиг вращения>.

Смещение 16, длина битового поля 8.

3.4. <смещение в таблице битовой подстановки>.

Смещение 24, длина битового поля 8.

3.5. <смещение в таблице байтовой подстановки>

Смещение 32, длина битового поля 8.

3.6. <закон изменения битовой маски>

Смещение 104 бита, длина битового поля 5 бит.

Значение данного поля интерпретируется следующим образом:

- **0 - <не задан>**
- **1 - <сложение>**
- **2 - <вращение>**
- **3 - <битовая подстановка>**
- **4 - <табличная подстановка>**

Остальные значения не присутствуют в выходном файле, а если присутствуют, то интерпретируются как <не задан>.

3.7. <параметр изменения битовой маски>.

Смещение 109 бит, длина битового поля 3 бита.

3.8. <таблица битовой подстановки>.

Смещение 2048, восемь элементов с длиной битового поля каждого 8 бит.

3.9. <таблица байтовой подстановки>.

Смещение 2112, 256 элементов с длиной битового поля каждого 8 бит.

4. Порядок применения алгоритмов.

Порядок применения алгоритмов <xor>, <add>, <rot>, <bts>, <tab>, <ecc>. Считывается очередной блок данных исходного файла, и к нему последовательно применяются вышеперечисленные алгоритмы, если же, конечно, они включены.

5. Используемые алгоритмы.

5.1. Парольная информация, хэширование пароля.

Для задания пароля используется опция командной строки --passwd, первая опция командной строки --passwd иммет дополнительный параметр вида: <пароль> | request, --passwd request рассматривается как указание программе запросить пароль с клавиатуры в диалоговом режиме, --passwd <пароль> указывает программе использовать строку <пароль>, заданную в командной строке в качестве пароля. Если в командной строке опция --passwd присутствует дважды, то вторая

опция рассматривается как --passwd <имя файла>. При этом <полный пароль> формируется следующим образом, к паролю заданному в явном виде или запрошенному с клавиатуры добавляется:

- **содержимое файла специфицированного второй опцией --passwd**
- **содержимое файла /dbin/<последнее имя файла>**
- **содержимое файла /usr/dbin/<последнее имя файла>**

где <последнее имя файла> выделяется из строки <имя файла>, заданного второй опцией --passwd, и представляет собой подстроку после последнего появления разделителя '/' в полном имени файла <имя файла>. Строки рассматриваются как строки ограниченные символом 0, и если содержимое файла содержит байт с нулевым значением, остаток файла начиная с нулевого байта и следующие файлы отбрасываются. Далее пароль копируется сам себе в конец до заполнения буфера длиной 1024 байт. Если пароль нулевой, пустая строка длиной 0 байт, то буфер заполняется нулями.

> **Примечание:** Если каталоги /dbin, /usr/dbin отсутствуют, то считается, что соответствующие парольные файлы пусты.

Далее буфер пароля подвергается операции хэширования:

- **если включен алгоритм <xor>, бит xor описанный выше, в подпункте 2.2, установлен, то каждый байт буфера подвергается операции xor с байтом <маска xor>, описанным в 3.1.**
- **если включен алгоритм <add>, бит add описанный в подпункте 2.3, установлен, то каждый байт буфера подвергается операции сложение с байтом <слагаемое>, описанным в 3.2.**
- **если включен алгоритм <rot>, бит rot описанный в подпункте 2.4, установлен, то каждый байт буфера подвергается операции врашение в сторону старших разрядов с битовым смещением <сдвиг вращения>, описанным в 3.3.**
- **если включен алгоритм <bts>, бит bts описанный в подпункте 2.5, установлен, то каждый байт буфера подвергается операции битовая подстановка.**

  Таблица битовой подстановки, состоящая из 8 байт формируется из таблицы <таблица битовой подстановки>, описанной в 3.8.

  Байт <смещение в таблице битовой подстановки>, описанный в 3.4, задаёт индекс нулевого элемента таблицы <таблица битовой подстановки> в результирующей таблице битовой подстановки. Т.е. в элемент <смещение в таблице битовой подстановки> результирующей таблицы заносится нулевой элемент таблицы <таблица битовой подстановки>. Затем, если результирующая таблица не заполнена, два индекса инкрементируются, и если индекс в результирующей таблице выходит за её пределы, то индекс заворачивается на нулевой, далее очередной байт копируется из исходной таблицы в результирующую таблицу, и эта операция повторяется до заполнения результирующей таблицы.

  Каждый байт исходного буфера подвергается операции битовая подстановка, если в исходном байте взведен бит <результирующая таблица битовой подстановки>[<номер элемента>], то в результирующем байте взводится бит <номер элемента>, где <номер элемента> представляет собой номер элемента в результирующей

таблице битовой подстановки, диапазон изменения от 0 до 7, а последовательность '[<номер элемента>]' обозначает извлечение элемента с номером <номер элемента> из таблицы. Если этот <результирующая таблица битовой подстановки>[<номер элемента>] бит исходного байта сброшен, то сбрасывается бит <номер элемента> в результирующем байте.

- **если включен алгоритм <tab>, бит tab, описанный в подпункте 2.6, установлен, то каждый байт буфера подвергается операции табличная подстановка.**

  Таблица байтовой подстановки, состоящая из 256 байт, формируется из таблицы <таблица байтовой подстановки>, описанной в 3.9. Байт <смещение в таблице байтовой подстановки>, описанный в 3.5, задаёт индекс нулевого элемента таблицы <таблица байтовой подстановки> в результирующей таблице. Те в элемент <смещение в таблице байтовой подстановки> результирующей таблицы заносится нулевой элемент таблицы <таблица байтовой подстановки>. Затем, если результирующая таблица не заполнена, два индекса инкрементируются, и если индекс в результирующей таблице выходит за её пределы, то индекс заворачивается на нулевой, далее очередной байт копируется из исходной таблицы в результирующую таблицу, и эта операция повторяется до заполнения результирующей таблицы.

  Далее к результирующей таблице байтовой подстановки применяются последовательно алгоритмы <xor>,<add>,<rot>,<bts>, т.е. если включен алгоритм <xor>, бит xor описанный выше, в подпункте 2.2, установлен, то каждый байт результирующей таблицы байтовой подстановки подвергается операции xor с байтом <маска xor>, описанным в 3.1, если включен алгоритм <add>, бит add описанный в подпункте 2.3, установлен, то каждый байт результирующей таблицы байтовой подстановки подвергается операции сложение с байтом <слагаемое>, описанным в 3.2, если включен алгоритм <rot>, бит rot описанный в подпункте 2.4, установлен, то каждый байт результирующей таблицы байтовой подстановки подвергается операции вращение в сторону старших разрядов с битовым смещением <сдвиг вращения>, описанным в 3.3, если включен алгоритм <bts>, бит bts описанный в подпункте 2.5, установлен, то каждый байт результирующей таблицы байтовой подстановки подвергается операции битовая подстановка, при этом порядок формирования результирующей таблицы битовой подстановки, и её применение описаны выше.

5.2. Используемые алгоритмы, <xor>.

Каждый байт исходного буфера подвергается операции исключающее или с битовой маской.

Стартовое значение битовой маски равно сумме параметра <маска xor>, описанного в пункте 3.1 и нулевого байта хэшированного пароля.

После каждой операции исключающее или битовая маска изменяется в соответствии с законом изменения битовой маски, который формируется как сумма параметра <закон изменения битовой маски>, описанного в 3.6 и 5и старших бит нулевого байта хэшированного пароля по модулю 5, т.е. применяемый закон изменения битовой маски равен (<закон изменения битовой маски> + (<хэшированный пароль>[0] >> 3))

% 5, где '>> 3' обозначает сдвиг вправо, в сторону младших разрядов на три разряда, а '% 5' - остаток от деления на 5.

- **Для закона изменения битовой маски - <не задан> битовая маска в процессе кодирования не изменяется.**
- **Для закона изменения битовой маски - <сложение> битовая маска в процессе кодирования инкрементируется на сумму <параметр изменения битовой маски>, описанного в пункте 3.7 и нулевого байта хэшированного пароля. Те величина инкрементирования маски операции составляет (<параметр изменения битовой маски>+<хэшированный пароль>[0]) & 255, где символ '+' обозначает операцию сложения, '[0]' обозначает нулевой элемент, '& 255' - операцию извлечения восьми последних битов суммы.**
- **Для закона изменения битовой маски - <вращение> битовая маска в процессе кодирования вращается в сторону старших разрядов на три младших бита суммы параметра <параметр изменения битовой маски>, описанного в пункте 3.7 и нулевого байта хэшированного пароля.**
- **Для закона изменения битовой маски <битовая подстановка> маска подвергается операции битовая подстановка после каждой операции исключающее или.**

  Таблица битовой подстановки, состоящая из 8 байт формируется из таблицы <таблица битовой подстановки>, описанной в 3.8.

  Три младших бита суммы параметра <параметр изменения битовой маски>, описанного в пункте 3.7 и нулевого байта хэшированного пароля, задают индекс нулевого элемента таблицы <таблица битовой подстановки> в результирующей таблице битовой подстановки, т.е. в элемент (<параметр изменения битовой маски> + <хэшированный пароль>[0]) & 7, где символ '+' обозначает операцию сложения, '[0]' обозначает нулевой элемент, '& 7' - операцию извлечения трех последних битов суммы, таблицы <результирующая таблица битовой подтановки> заносится нулевой элемент таблицы <таблица битовой подстановки>. Затем, если таблица не заполнена, два индекса инкрементируются, и если индекс в результирующей таблице выходит за её пределы, то индекс заворачивается на нулевой, далее байт копируется из исходной таблицы в результирующую таблицу, и эта операция повторяется до заполнения результирующей таблицы.

  Если в байте битовой маски взведен бит <результирующая таблица битовой подстановки>[<номер элемента>], то в результирующем байте маски взводится бит <номер элемента>, где <номер элемента> представляет собой номер элемента в результирующей таблице битовой подстановки, диапазон изменения от 0 до 7, а последовательность [<номер элемента>] обозначает извлечение элемента с номером <номер элемента> из таблицы. Если <результирующая таблица битовой подстановки>[<номер элемента>] бит маски сброшен, то сбрасывается бит <номер элемента> в результирующей маске.

- **Для закона изменения битовой маски <табличная подстановка>, битовая маска подвергается операции табличная подстановка после каждой операции исключающее или.**

  Таблица байтовой подстановки, состоящая из 256 байт формируется из таблицы <таблица байтовой подстановки>, описанной в 3.9.

  Сумма параметра <параметр изменения битовой маски>, описанного в пункте 3.7 и нулевого байта хэшированного пароля, задают индекс

нулевого элемента таблицы <таблица байтовой подстановки> в результирующей таблице. Те в элемент (<параметр изменения битовой маски> + <хэшированный пароль>[0]) & 255 результирующей таблицы заносится нулевой элемент таблицы <таблица байтовой подстановки>, где '[0]' обозначает процедуру извлечения нулевого элемента, а '& 255' обозначает процедуру извлечения восьми младших бит. Затем, если таблица не заполнена, два индекса инкрементируются, и если индекс в результирующей таблице выходит за её пределы, то индекс заворачивается на нулевой, далее байт копируется из исходной таблицы в результирующую таблицу, и эта операция повторяется до заполнения результирующей таблицы.

5.3. Используемые алгоритмы, <add>.

<слагаемое>, описанное в пункте 3.2, инкрементируется на первый байт хэшированного пароля, затем каждый байт исходного буфера инкрементируется на полученное значение.

5.4. Используемые алгоритмы, <rot>.

<сдвиг вращения>, описанный в пункте 3.3, инкрементируется на второй байт хэшированного пароля, затем каждый байт исходного буфера вращается в сторону наиболее значимых разрядов на полученное значение.

5.5. Используемые алгоритмы, <bts>.

Каждый байт входного блока данных подвергается операции битовая подстановка.

Таблица битовой подстановки, состоящая из 8 байт формируется из таблицы <таблица битовй подстановки>, описанной в 3.8.

Три младших бита суммы параметра <смещение в таблице битовой подстановки>, описанного в 3.4 и третьего элемента хэшированного пароля задают индекс нулевого элемента таблицы <таблица битовой подстановки> в результирующей таблице битовой подстановки. Те в элемент (<смещение в таблице битовой подстановки> + <хэшированный пароль>[3]) & 7 результирующей таблицы заносится нулевой элемент таблицы <таблица битовой подстановки>, где '[3]' обозначает извлечение третьего элемента, а '& 7' обозначает извлечение трех младших битов. Затем, если таблица не заполнена, два индекса инкрементируются, и если индекс в результирующей таблице выходит за её пределы, то индекс заворачивается на нулевой, далее байт копируется из исходной таблицы в результирующую таблицу, и эта операция повторяется до заполнения результирующей таблицы.

Каждый байт исходного буфера подвергается операции битовая подстановка, если в исходном байте взведен бит <результирующая таблица битовой подстановки>[<номер элемента>] то в результирующем байте взводится бит <номер элемента>, где <номер элемента> представляет собой номер элемента в результирующей таблице битовой подстановки, диапазон изменения от 0 до 7, а последовательность [<номер элемента>] обозначает извлечение элемента с номером <номер элемента> из таблицы. Если <результирующая таблица битовой подстановки>[<номер элемента>]

бит исходного байта сброшен, то сбрасывается бит <номер элемента> в результирующем байте.

5.6. Используемые алгоритмы, <tab>.

Каждый байт исходного буфера подвергается операции табличная подстановка. После каждого такого преобразования входного буфера, таблица байтовой подстановки модифицируется в соотвествии с очередным полубайтом хэшированного пароля, полубайты извлекаются сначала старший полубайт, потом младший, сначала из байта с младшим номером, а потом из старшего. Всего используется 256 байта хэшированного пароля. Процедура формирования стартовой таблицы приведена после описания процедуры модификации таблицы подстановки байтов.

Если взведён бит 0 полубайта, и взведён бит включения алгоритма <xor>, описанный в 2.2, то текущая таблица байтовой подстановки подвергается операции исключающее или с параметром <маска xor>, описанным в 3.1.

Если взведен бит 1 полубайта, и взведён бит включения алгоритма <add>, описанный 2.3, то текущая таблица байтовой подстановки подвергается операции сложение с параметром <слагаемое>, описанным в 3.2.

Если взведен бит 2 полубайта, и взведён бит включения алгоритма <rot>, описанный 2.4, то таблица байтовой подстановки подвергается операции вращение в сторону старших разрядов со сдвигом <сдвиг вращения>, описанным в 3.3.

Если взведен бит 3 полубайта, и взведён бит включения алгоритма <bts>, описанный 2.5, то таблица байтовой подстановки подвергается операции битовая подстановка.

Таблица битовой подстановки, состоящая из 8 байт формируется из таблицы <таблица битовой подстановки>, описанной в 3.8. Байт <смещение в таблице битовой подстановки>, описанный в 3.4 задаёт индекс нулевого элемента таблицы <таблица битовой подстановки> в результирующей таблице битовой подстановки. Те в элемент <смещение в таблице битовой подстановки> результирующей таблицы заносится нулевой элемент таблицы <таблица битовой подстановки>. Затем, если таблица не заполнена, два индекса инкрементируются, и если индекс в результирующей таблице выходит за её пределы, то индекс заворачивается на нулевой, далее байт копируется из исходной таблицы в результирующую таблицу, и эта операция повторяется до заполнения результирующей таблицы. Каждый байт таблицы подвергается операции битовая подстановка, если в исходном байте взведен бит <результирующая таблица битовой подстановки>[<номер элемента>], то в результирующем байте взводится бит <номер элемента>, где <номер элемента> представляет собой номер элемента в результирующей таблице битовой подстановки, диапазон изменения от 0 до 7, а последовательность [<номер элемента>] обозначает извлечение элемента с номером <номер элемента> из таблицы. Если <результирующая таблица битовой подстановки>[<номер элемента>] бит исходного байта сброшен, то сбрасывается бит <номер элемента> в результирующем байте.

Стартовая таблица байтовой подстановки, состоящая из 256 байт формируется из таблицы <таблица байтовой подстановки>, описанной в 3.9. 8 младших битов суммы параметра <смещение в таблице байтовой подстановки>, описанного в 3.5 и четвертого байта хэшированного пароля, задают индекс нулевого элемента таблицы <таблица байтовой подстановки> в результирующей таблице. Те в элемент

(<смещение в таблице байтовой подстановки> + <хэшированный пароль>[4]) & 255 результирующей таблицы заносится нулевой элемент таблицы <таблица байтовой подстановки>, где '[4]' обозначает извлечение четвертого элемента, а (& 255) обозначает извлечение 8 младших битов суммы. Затем, если таблица не заполнена, два индекса инкрементируются, и если индекс в результирующей таблице выходит за её пределы, то индекс заворачивается на нулевой, далее байт копируется из исходной таблицы в результирующую таблицу, и эта операция повторяется до заполнения результирующей таблицы.

5.7. Используемые алгоритмы, <ecc>.

Исходный блок данных разбивается на подблоки по 256 байт, последний полученный таким образом подблок данных, если он не полный дописывается до 256 байт байтом со значением 255, для полученных таким образом подблоков данных считаются три исправляющих байта, которые добавляются в конец каждого исходного подблока, полного или неполного. Затем, полученные таким образом подблоки записываются в выходной файл, начиная со стартового подблока данных. Алгоритм расчета исправляющих байтов аналогичен используемому в Linux [1] версии 2.4.

Приложение A. Лицезия

Лицензия программы ucrp 1.30

электронная почта: dch@ucrouter.ru, divch@users.sourceforge.net, dch@ucrouter.com; телефон: +7-916-315-1943; ICQ: 474-363-900;

Лицензия усечённой/бесплатной версии программы

#endif

Если Вы имеете исходные тексты программы ucrp с текстовой строкой __UCRP_PRIVATE__ внутри файлов, Вы не можете использовать и распространять данные файлы без устного или письменного разрешения выше упомянутого автора.

Вы можете удалить код расширения ucrp и использовать и распространять эти файлы как свободное программное обеспечение под данной лицензией.

Приложение B. Опции командной строки

--license	**выдача лицензии программы, язык английский**
--license-ru	**выдача лицензии программы на русском языке**
--encrypt	**криптование архива**
--decrypt	**декриптование архива**
--passwd <passwd> | request	**<passwd> задание пароля в явном виде, request - указание программе запросить пароль с клавиатуры**
--hide-passwd [*|$|d]	**скрыть пароль при запросе с клавиатуры, дополнительный параметр символ который печатается на устройство вывода вместо введённого символа**
--show-passwd	**показать пароль при клавиатурном вводе**
--history	**напечатать историю вызовов программы и закончить работу** *(отсутствует в усечённой/бесплатной версии программы)*
--history-clear	**очистка файла истории** *(отсутствует в усечённой/бесплатной версии программы)*
--history-encrypt	**зашифровать файл истории** *(отсутствует в усечённой/бесплатной версии программы)*
--history-decrypt	**расшифровать файл истории** *(отсутствует в усечённой/бесплатной версии программы)*
--history-cat	**вывести на устройство вывода файл истории** *(отсутствует в усечённой/бесплатной версии программы)*
--block-size	**задаёт размер блока данных**
--change-dir	**изменение текущего каталога**
--file-passwd <имя файла>	**запрос паролей с клавиатуры и запись паролей в файлы <имя файла>, /dbin/<последнее имя файла>, /usr/dbin/<последнее имя файла>, <последнее имя файл> подстрока строки <имя файла> от последнего разделителя каталогов.** *(отсутствует в усечённой/бесплатной версии программы)*
--master	**режим TCP/IP сервера. Запрашивает парольную информацию с клавиатуры, отключается от клавиатуры, выдает клиентам по TCP/IP соединению запрошенную парольную информацию.** *(отсутствует в усечённой/бесплатной версии программы)*

--connect [p][f]	**режим клиента, запрашивает пароль и имя парольного файла, заданного опцией --passwd. p - запрос пароля у сервера, f - запрос имени файла с парольной информацией. При отстутвии дополнительных параметров запрашивается пароль с именем заданным первой опцией --passwd и имя парольного файла, заданного второй опцией --passwd, если вторая опция --passwd отсутствует, запрашивается имя парольного файла заданного первой опцией --passwd.** *(отсутствует в усечённой/бесплатной версии программы)*
--connect-test	**тестирование клиент/серверного соединения** *(отсутствует в усечённой/бесплатной версии программы)*
--terminate	**завершение сервера** *(отсутствует в усечённой/бесплатной версии программы)*
--info	**печать информации о зашифрованном файле**
--clear [inc|0,1]	**очистка свободного дискового пространства, bash, mc истории, Kicad-овских логов, inc запись инкрементируемого байта, 0 - запись 0, 1 - запись байта 255**
--passwd-files-encrypt	**криптование парольных файлов** *(отсутствует в усечённой/бесплатной версии программы)*
--passwd-files-decrypt	**декриптование парольных файлов** *(отсутствует в усечённой/бесплатной версии программы)*
--passwd-files-clear	**очистка парольных файлов** *(отсутствует в усечённой/бесплатной версии программы)*
--archives	**количество архивных файлов** *(отсутствует в усечённой/бесплатной версии программы)*
--test	**тестирование программы, генерируется случайный файл, который последовательно шифруется, дешифруется, сравнивается с оригинальным при заданном пароле, при обнаружении ошибки выдается сообщение о ошибке и программа завершает свою работу.** *(отсутствует в усечённой/бесплатной версии программы)*
--private <макро>	**исключение конструкций #ifdef <макро> #else #endif из файлов исходных текстов** *(отсутствует в усечённой/бесплатной версии программы)*
--disable xor | add | rot | tab | ecc | bts | xre | std	**отключение алгоритмов**
--enable xor | add | rot | tab | ecc | bts | xre | std	**включение алгоритмов**
--quiet	**ничего не выдавать на стандартное устройство вывода**
--version	**печать версии программы**
--todo	**печать комментария автора "что надо сделать"**
--changelog	**выдача журнала изменений**
--man	**выдача справочной страницы**
--unreleased	**выдача нереализованных особенностей** *(отсутствует в усечённой/бесплатной версии программы)*

--help	**выдача справки по опциям**

Notes

1. **Linux® зарегистрированная торговая марка Linus Torvalds.**