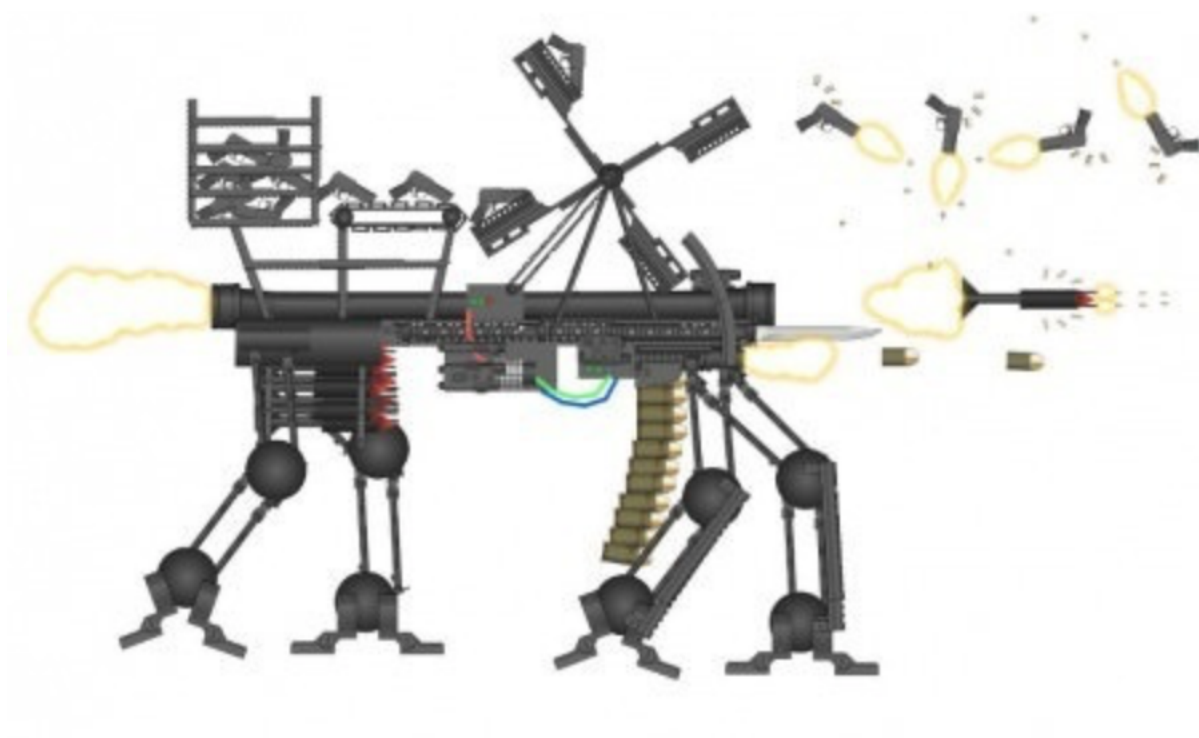


Ф.В. Леонов, А.Д. Челпанов, А.Н. Югансон

**ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ PYTHON ДЛЯ
РЕШЕНИЯ ЗАДАЧ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ.
МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ
ПРАКТИЧЕСКИХ РАБОТ**



**Санкт-Петербург
2021**

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

Ф.В. Леонов, А.Д. Челпанов, А.Н. Югансон
ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ PYTHON ДЛЯ
РЕШЕНИЯ ЗАДАЧ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ.
МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ
ПРАКТИЧЕСКИХ РАБОТ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

РЕКОМЕНДОВАНО К ИСПОЛЬЗОВАНИЮ В УНИВЕРСИТЕТЕ
ИТМО

по направлению подготовки 10.04.01 Информационная
безопасность

в качестве Учебно-методического пособия для реализации основных
профессиональных образовательных программ высшего образования
магистратуры

 УНИВЕРСИТЕТ ИТМО

Санкт-Петербург
2021

Леонов Ф.В., Челпанов А.Д., Югансон А.Н., Программирование на языке Python для решения задач информационной безопасности. Методические указания по выполнению практических работ.– СПб: Университет ИТМО, 2021. – 36 с.

Рецензент(ы):

Кузнецов Александр Юрьевич, кандидат технических наук, доцент (квалификационная категория "ординарный доцент") факультета безопасности информационных технологий, Университета ИТМО.

Учебно-методическое пособие предназначено для студентов по направлению подготовки 10.04.01 «Информационная безопасность» в качестве учебного пособия для выполнения практических работ по курсу «Программирование на языке Python для решения задач информационной безопасности»



Университет ИТМО – ведущий вуз России в области информационных и фотонных технологий, один из немногих российских вузов, получивших в 2009 году статус национального исследовательского университета. С 2013 года Университет ИТМО – участник программы повышения конкурентоспособности российских университетов среди ведущих мировых научно-образовательных центров, известной как проект «5 в 100». Цель Университета ИТМО – становление исследовательского университета мирового уровня, предпринимательского по типу, ориентированного на интернационализацию всех направлений деятельности.

© Университет ИТМО, 2021

© Леонов Ф.В., Челпанов А.Д., Югансон А.Н., 2021

Содержание

| | |
|-----------------------------------------------------------------------------------------------------------|----|
| Введение..... | 4 |
| Практическая работа №1 «Решение простых алгоритмических задач»..... | 5 |
| Практическая работа №2 «Поиск информации в массивах данных по заданным критериям» | 13 |
| Практическая работа №3 «Парсинг двоичных файлов различного формата» | 18 |
| Практическая работа №4 «Реализация системы удаленного выполнения команд с использованием шифрования»..... | 25 |
| Список литературы | 30 |
| Приложение 1. Реализация протокола Диффи-Хеллмана..... | 32 |
| Приложение 2. Реализация алгоритма AES | 33 |

Введение

Python по-прежнему является доминирующим языком в мире информационной безопасности. Если вы заинтересованы в решении задач информационной безопасности, Python - отличный язык для изучения из-за большого количества библиотек для реверс-инжиниринга и эксплуатации, доступных для использования.

Методические указания охватывают широкий круг тем, освоение которых позволит обучающимся создавать приложения для решения специфических задач. А как известно, разница между script kiddies и профессионалами - это разница между простым использованием чужих инструментов и написанием собственных.

Учебно-методическое пособие предназначено для студентов магистратуры по направлению подготовки 10.04.01 «Информационная безопасность» в качестве учебного пособия для выполнения практических работ по курсу «Программирование на языке Python для решения задач информационной безопасности» с целью получения навыков работы с основными типами и структурами данных, операторами, а также развития компетенций по работе с файловой системой, окружением и процессами.

Практическая работа №1 «Решение простых алгоритмических задач»

Цель работы: знакомство с основными конструкциями языка Python и получение навыков их применения для создания алгоритмов автоматизации действий при работе с файловой системой.

Основные конструкции языка

– типы данных:

– числа (int, float):

```
In [1]: type(1)
Out[1]: int

In [2]: type(10/3)
Out[2]: float

In [3]: int_digit = 10

In [4]: int_digit ** 3
Out[4]: 1000

In [5]: int_digit // 3
Out[5]: 3

In [6]: int_digit % 3
Out[6]: 1
```

– строки (str):

```
In [1]: s = "string"

In [2]: type(s)
Out[2]: str

In [3]: s
Out[3]: 'string'

In [4]: len(s)
Out[4]: 6

In [5]: "__" + s + "--"
Out[5]: '__string--'

In [6]: s.find("i")
Out[6]: 3

In [7]: s.upper()
Out[7]: 'STRING'
```

– логические значения (bool):

```
In [1]: type(True)
Out[1]: bool

In [2]: type(False)
Out[2]: bool

In [3]: True or False
Out[3]: True
```

```
In [4]: True and False
Out[4]: False

In [5]: a = False

In [6]: a is False
Out[7]: True

In [8]: a is not True
Out[8]: True
```

– структуры данных:

– списки (list) – это изменяемые последовательности, обычно используемые для хранения коллекций однородных элементов

```
In [1]: xs = [0,1,2,3]

In [2]: type(xs)
Out[2]: list

In [3]: xs * 3
Out[3]: [0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3]

In [4]: len(xs)
Out[4]: 4

In [5]: xs[2]=0

In [6]: xs
Out[6]: [0, 1, 0, 3]

In [7]: xs.append(4)

In [8]: xs
Out[8]: [0, 1, 0, 3, 4]

In [9]: del xs[0]

In [10]: xs
Out[10]: [1, 0, 3, 4]

In [11]: xs.extend("abc")

In [12]: xs
Out[12]: [1, 0, 3, 4, 'a', 'b', 'c']
```

– кортежи (tuples) – это неизменяемые последовательности, обычно используемые для хранения коллекций разнородных данных

```
In [1]: coordinate = (0,3)

In [2]: type(coordinate)
Out[2]: tuple

In [3]: coordinate[0]
Out[3]: 0

In [4]: len(coordinate)
Out[4]: 2
```

```
In [5]: coordinate[0]=5
```

```
-----  
TypeError                                 Traceback (most recent call  
last)  
<ipython-input-5-7a6e08c43709> in <module>  
----> 1 coordinate[0]=5
```

TypeError: 'tuple' object does not support item assignment

– диапазон (`range`) – это неизменяемая последовательность чисел

```
In [1]: type(range(10))
```

```
Out[1]: range
```

```
In [2]: range(10)
```

```
Out[2]: range(0, 10)
```

```
In [3]: list(range(10))
```

```
Out[3]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

– множество (`set`) – это неупорядоченная коллекция без повторяющихся элементов

```
In [1]: basket = {"a", "b", "c", "d", "e", "d", "c", "b", "a"}
```

```
In [2]: type(basket)
```

```
Out[2]: set
```

```
In [3]: len(basket)
```

```
Out[3]: 5
```

```
In [4]: basket.add("f")
```

```
In [5]: basket
```

```
Out[5]: {'a', 'b', 'c', 'd', 'e', 'f'}
```

```
In [6]: basket.discard("b")
```

```
In [7]: basket
```

```
Out[7]: {'a', 'c', 'd', 'e', 'f'}
```

– словарь (`dict`) – это набор пар вида «ключ: значение» с требованием, чтобы ключи были уникальными (в пределах одного словаря) и неизменяемыми

```
In [1]: ages = {"Alisa": 19, "Bob": 25}
```

```
In [2]: type(ages)
```

```
Out[2]: dict
```

```
In [3]: ages
```

```
Out[3]: {'Alisa': 19, 'Bob': 25}
```

```
In [4]: ages["Bob"]
```

```
Out[4]: 25
```

```
In [5]: ages["Alisa"] = 20
```

```
In [6]: ages
```

```
Out[6]: {'Alisa': 20, 'Bob': 25}
```

```
In [7]: ages.keys()
```



```
Out[7]: dict_keys(['Alisa', 'Bob'])
In [8]: ages.items()
Out[8]: dict_items([('Alisa', 20), ('Bob', 25)])
```

– операторы:

– `if` используется для условного выполнения

```
In [1]: a = 1703

In [2]: if a % 2 == 0:
...:     a /= 2
...: elif a % 5 == 0:
...:     a /= 5
...: else:
...:     a = 0
...:
```

```
In [3]: a
Out[3]: 0
```

– `for` выполняет итерацию по элементам любой последовательности (список или строка) в том порядке, в котором они появляются в последовательности:

```
In [1]: for ch in 'some_string':
...:     print(ch, end = ' ')
...:
s o m e _ s t r i n g
```

– `while` используется для повторного выполнения операций, пока выражение истинно:

```
In [1]: i = 0

In [2]: while i < 10:
...:     print(i, end = ' ')
...:     i += 1
...:
0 1 2 3 4 5 6 7 8 9
```

Встроенные библиотеки

1. `os`¹ – этот модуль предназначен для работы с операционной системой. Он позволяет работать с файловой системой, окружением, управлять процессами, создавать каталоги и проверять существование файлов и директорий.
2. `subprocess`² – этот модуль позволяет создавать новые процессы и управлять их работой:
 - подключаться к стандартным потокам ввода/вывода нового процесса;
 - получать код возврата;
 - ожидать завершения процесса;
 - использовать механизмы межпроцессного взаимодействия IPC.

¹ `os` — Miscellaneous operating system interfaces, [Электронный ресурс]. URL: <https://docs.python.org/3/library/os.html> (дата обр. 28.02.2021)

² `argparse` — Parser for command-line options, arguments and sub-commands, [Электронный ресурс]. URL: <https://docs.python.org/3/library/argparse.html> (дата обр. 28.02.2021)

С помощью subprocess можно, например, выполнять любые команды Linux из скрипта и, в зависимости от ситуации, получать вывод или только проверять, что команда выполнена без ошибок.

3. `argparse`³ – это модуль для обработки аргументов командной строки.

Он позволяет:

- создавать аргументы и опции, с которыми может вызываться скрипт;
- указывать типы аргументов, значения по умолчанию;
- указывать, какие действия соответствуют аргументам;
- выполнять вызов функции при указании аргумента;
- отображать сообщения с подсказками по использованию программы.

Описание задач

1. Поиск файла

- Задание: осуществить поиск файла с указанным именем в заданной директории.
- Параметры программы:
 - `-r`: рекурсивный поиск;
 - `-d <число>`: максимальная глубина поиска.
- Входные данные:
 - имя файла;
 - путь до директории.
- Выходные данные:
 - путь до файла. Если было найдено несколько путей – вывести любой из них.

2. Поиск файлов, имеющих определенный тип

- Задание: осуществить поиск файлов, имеющих указанный тип в заданной директории.
- Параметры программы:
 - `-r`: рекурсивный поиск;
 - `-d <число>`: максимальная глубина поиска.
- Входные данные:
 - тип файла;
 - путь до директории.
- Выходные данные:
 - список файлов.

3. Поиск файлов, содержащих определенный текст

³ subprocess — Subprocess management, [Электронный ресурс]. URL: <https://docs.python.org/3/library/subprocess.html> (дата обр. 28.02.2021)

- Задание: осуществить поиск файлов, в которых имеется указанный текст в заданной директории.
 - Параметры программы:
 - -r: рекурсивный поиск;
 - -d <число>: максимальная глубина поиска;
 - -c не учитывать регистр символов.
 - Входные данные:
 - текст;
 - путь до директории.
 - Выходные данные:
 - список файлов.
4. Поиск файлов с определенным размером
- Задание: осуществить поиск файлов, которые соответствуют указанному размеру в заданной директории.
 - Параметры программы:
 - -r: рекурсивный поиск;
 - -g: искать файлы с большим размером, чем аргумент;
 - -l: искать файлы с меньшим размером, чем аргумент;
 - -s: отсортировать файлы по возрастанию размера.
 - Входные данные:
 - размер в байтах;
 - путь до директории.
 - Выходные данные:
 - список файлов и соответствующие им размеры.
5. Поиск файлов с определенной датой модификации
- Задание: осуществить поиск файлов, которые соответствуют указанной дате модификации в заданной директории.
 - Параметры программы:
 - -r: рекурсивный поиск;
 - -g: искать файлы более старшей датой модификации, чем аргумент;
 - -l: искать файлы более ранней датой модификации, чем аргумент;
 - -s: отсортировать файлы по возрастанию даты модификации.
 - Входные данные:
 - размер в байтах;
 - путь до директории.
 - Выходные данные:
 - список файлов и соответствующие им даты модификации.

6. Проверка числа на принадлежность к множеству Фибоначчи
 - Задание: осуществить проверку, входит ли число в множество чисел Фибоначчи.
 - Параметры программы:
 - Входные данные:
 - целое число.
 - Выходные данные: True/False.
7. Получение произвольного числа Фибоначчи
 - Задание: вычислить число Фибоначчи по его порядковому номеру.
 - Параметры программы:
 - Входные данные:
 - целое число.
 - Выходные данные:
 - число.
8. Проверка числа на простоту методом Ферма
 - Задание: провести тест Ферма для заданного целого числа.
 - Параметры программы:
 - Входные данные:
 - целое число.
 - Выходные данные:
 - True/False.
9. Проверка числа на простоту методом Миллера-Рабина
 - Задание: провести тест Миллера-Рабина для заданного целого числа.
 - Параметры программы:
 - -r <число>: количество раундов.
 - Входные данные:
 - целое число.
 - Выходные данные:
 - True/False.
10. Получение значения функции Эйлера для произвольного числа
 - Задание: вычислить значение функции Эйлера для заданного целого числа.
 - Параметры программы:
 - Входные данные:
 - целое число.
 - Выходные данные:
 - число.
11. Получение уникальных строк из текста

- Задание: получить список уникальных строк, которые встречаются в переданном тексте.
- Параметры программы:
 - -s: отсортировать уникальные строки по возрастанию.
- Входные данные:
 - строка.
- Выходные данные:
 - список уникальных строк.

12. Создание словаря паролей на основе утечки

- Задание: на основе файла утечки формата CSV создать словарь наиболее используемых паролей.
- Параметры программы:
 - -n <число>: размер словаря;
 - -s : отсортировать в порядке убывания вхождений.
- Входные данные:
 - путь до файла утечки.
- Выходные данные:
 - словарь в виде "пароль: количество вхождений".

Варианты

| № варианта | Список задач | № варианта | Список задач |
|------------|--------------------|------------|--------------------|
| 1 | 3, 4, 6, 9, 10, 12 | 11 | 2, 4, 7, 9, 10, 12 |
| 2 | 1, 4, 6, 8, 11, 12 | 12 | 2, 5, 6, 9, 11, 12 |
| 3 | 2, 4, 7, 8, 11, 12 | 13 | 3, 5, 7, 8, 11, 12 |
| 4 | 2, 5, 7, 8, 11, 12 | 14 | 2, 4, 7, 8, 10, 12 |
| 5 | 1, 5, 7, 8, 10, 12 | 15 | 2, 4, 6, 8, 11, 12 |
| 6 | 3, 4, 7, 8, 11, 12 | 16 | 1, 4, 6, 9, 10, 12 |
| 7 | 3, 5, 6, 8, 11, 12 | 17 | 3, 4, 7, 9, 10, 12 |
| 8 | 2, 5, 6, 9, 10, 12 | 18 | 1, 5, 6, 9, 11, 12 |
| 9 | 3, 4, 7, 8, 10, 12 | 19 | 1, 4, 6, 9, 11, 12 |
| 10 | 1, 5, 7, 9, 10, 12 | 20 | 2, 4, 6, 9, 10, 12 |

Практическая работа №2 «Поиск информации в массивах данных по заданным критериям»

Цель работы: знакомство с регулярными выражениями и получение навыков их применения для поиска и валидации определенной информации в массивах данных.

Регулярные выражения

Регулярные выражения – формальный язык для работы с подстроками в тексте, основанный на использовании метасимволов.

Квантификатор используется для обозначения количества повторений части регулярного выражения:

| Квантификатор | Назначение |
|----------------|------------------------------------------|
| a{n} | ровно n раз подряд «а» |
| a{n, m} | между n и m раз подряд «а» |
| a{n,} | n и больше раз подряд «а» |
| a? | 0 или 1 раз подряд «а» |
| a+ | 1 или больше раз подряд «а» |
| a* | 0 или больше раз подряд «а» |

Существуют метасимволы для часто встречающихся наборов:

| Метасимвол | Соответствующий класс символов |
|------------|--------------------------------|
| \d | [0-9] |
| \D | [^0-9] |
| \s | [\t\n\r\f\v] |
| \S | [^\t\n\r\f\v] |
| \w | [a-zA-Z0-9_] |
| \W | [^a-zA-Z0-9_] |

Якори используются для нахождения определенных границ в тексте и не охватывают никаких символов:

| Якорь | Назначение |
|-----------|--------------------------------------------------------------------------------|
| ^ | обозначает начало строки |
| \$ | обозначает конец строки |
| \b | обозначает конец слова |
| \B | обозначает позицию между двумя словами, соответствующими метасимволу \w |

Флаги используются для задания определенных параметров поиска:

| Флаг | Название | Назначение |
|----------|-------------|-----------------------------------------------------------------------------|
| s | single line | метасимвол . также учитывает символы переноса строк |
| g | global | поиск всех соответствий |
| m | multiline | метасимволы ^ и \$ применяются ко всем строчкам в отдельности |
| i | ignore case | при поиске выражения не учитывать регистр букв |
| x | verbose | поставляет оставлять комментарии через # ; |

| Флаг | Название | Назначение |
|------|----------|--------------------|
| | | игнорирует пробелы |

Встроенные библиотеки

1. `re`⁴ – этот модуль предоставляет операции сопоставления регулярных выражений: функции в этом модуле позволяют проверить, соответствует ли конкретная строка заданному регулярному выражению (или соответствует ли данное регулярное выражение определенной строке, что сводится к тому же самому).

Описание задач

1. Поиск csrf-токена
 - Задание: найти csrf-токен на экземплярах одной html-страницы.
 - Параметры программы:
 - Входные данные:
 - файл с html-страницей.
 - Выходные данные:
 - csrf-токен.
2. Поиск ссылок на изображения
 - Задание: найти все ссылки на используемые на html-странице изображения.
 - Параметры программы:
 - Входные данные:
 - файл с html-страницей.
 - Выходные данные:
 - список уникальных ссылок.
3. Поиск ссылок
 - Задание: найти все ссылки на одной html-странице.
 - Параметры программы:
 - Входные данные:
 - файл с html-страницей.
 - Выходные данные:
 - список уникальных ссылок.
4. Валидация ссылки
 - Задание: определить, является ли переданная строка корректной ссылкой.
 - Параметры программы:
 - Входные данные:
 - строка.
 - Выходные данные:
 - True/False.

⁴ re — Regular expression operations, [Электронный ресурс]. URL: <https://docs.python.org/3/library/re.html> (дата обр. 28.02.2021)

5. Поиск email адресов

- Задание: найти все email адреса в заданном тексте.
- Параметры программы:
- Входные данные:
 - текстовый файл.
- Выходные данные:
 - список уникальных email адресов.

6. Валидация email адреса

- Задание: определить, является ли переданный email адрес корректным.
- Параметры программы:
- Входные данные:
 - строка.
- Выходные данные:
 - True/False.

7. Поиск номеров телефонов

- Задание: найти все телефонные номера в заданном тексте. Номер телефона может содержать:
 - пробелы для разделения кода оператора;
 - скобочки для выделения кода оператора;
 - символ "+" в самом начале.
- Параметры программы:
 - `--lang <строка>`: страна регистрации мобильного телефона (обеспечить поддержку российского (ru), американского (usa), белорусского (bel), китайского (ch) форматов).
- Входные данные:
 - текстовый файл.
- Выходные данные:
 - список уникальных номеров телефонов.

8. Валидация номеров телефонов

- Задание: определить является ли переданный телефонный номер корректным. Обеспечить поддержку российского, канадского, чешского, финского форматов. Номер телефона может содержать:
 - пробелы для разделения кода оператора;
 - скобочки для выделения кода оператора;
 - символ "+" в самом начале.
- Параметры программы:
- Входные данные:
 - текстовый файл.
- Выходные данные:

- True/False.

9. Поиск IPv4 адресов

- Задание: найти все IPv4 адреса в заданном тексте.
- Параметры программы:
- Входные данные:
 - текстовый файл.
- Выходные данные:
 - список уникальных IPv4 адресов.

10. Валидация IPv4 адреса

- Задание: определить, является ли переданная строка корректным IPv4 адресом.
- Параметры программы:
- Входные данные:
 - строка.
- Выходные данные:
 - True/False.

11. Исправление пробелов

- Задание: в заданном тексте необходимо навести порядок: один пробел между словами, два – между предложениями.
- Параметры программы:
- Входные данные:
 - строка.
- Выходные данные:
 - отформатированный текст.

12. Поиск дат, соответствующих формату

- Задание: в заданном тексте необходимо найти все даты, которые соответствуют формату (DD/MM/YYYY HH:MM:SS).
- Параметры программы:
 - -y <число>: год, по которому осуществляется фильтрация;
 - -n: даты, новее переданного года (2015 год новее 2011 год);
 - -o: даты, позднее переданного года (2010 год позднее 2011 год);
Если год совпадает, то он не включается в список ни с параметром -n, ни с -o.
- Входные данные:
 - текстовый файл.
- Выходные данные:
 - список дат.

Варианты

| № варианта | Список задач | № варианта | Список задач |
|-------------------|---------------------|-------------------|---------------------|
| 1 | 1, 2, 4, 5, 6, 11 | 11 | 1, 5, 6, 7, 8, 11 |

| № варианта | Список задач | № варианта | Список задач |
|-------------------|---------------------|-------------------|---------------------|
| 2 | 1, 2, 4, 7, 8, 12 | 12 | 1, 5, 6, 9, 10, 12 |
| 3 | 2, 4, 5, 6, 7, 8 | 13 | 5, 6, 7, 8, 9, 10 |
| 4 | 2, 4, 5, 6, 9, 10 | 14 | 5, 6, 7, 8, 11, 12 |
| 5 | 2, 4, 5, 6, 11, 12 | 15 | 5, 6, 9, 10, 11, 12 |
| 6 | 1, 3, 4, 5, 6, 11 | 16 | 1, 7, 8, 9, 10, 11 |
| 7 | 1, 3, 4, 7, 8, 12 | 17 | 7, 8, 9, 10, 11, 12 |
| 8 | 3, 4, 5, 6, 7, 8 | 18 | 1, 2, 4, 9, 10, 12 |
| 9 | 3, 4, 5, 6, 9, 10 | 19 | 1, 3, 4, 9, 10, 11 |
| 10 | 3, 4, 5, 6, 11, 12 | 20 | 2, 4, 7, 8, 9, 10 |

Практическая работа №3 «Парсинг двоичных файлов различного формата»

Цель работы: знакомство с двоичными файлами и получение навыков распаковки данных из заданной последовательности байтов.

Описание двоичных файлов

- Executable and Linkable Format, ELF⁵ – формат двоичных файлов, используемый во многих современных UNIX-подобных операционных системах.

Структура заголовка файла:

```
typedef struct {
    unsigned char e_ident[16]; /* Magic number and other info */
    uint16_t e_type; /* Object file type */
    uint16_t e_machine; /* Architecture */
    uint32_t e_version; /* Object file version */
    uint64_t e_entry; /* Entry point virtual address */
    uint64_t e_phoff; /* Program header table file offset */
    uint64_t e_shoff; /* Section header table file offset */
    uint32_t e_flags; /* Processor-specific flags */
    uint16_t e_ehsize; /* ELF header size in bytes */
    uint16_t e_phentsize; /* Program header table entry size */
    uint16_t e_phnum; /* Program header table entry count */
    uint16_t e_shentsize; /* Section header table entry size */
    uint16_t e_shnum; /* Section header table entry count */
    uint16_t e_shstrndx; /* Section header string table index */
} Elf64_Ehdr;
```

Структура заголовков секций:

```
typedef struct {
    uint32_t sh_name; /* Section name (string tbl index) */
    uint32_t sh_type; /* Section type */
    uint64_t sh_flags; /* Section flags */
    uint64_t sh_addr; /* Section virtual addr at execution */
    uint64_t sh_offset; /* Section file offset */
    uint64_t sh_size; /* Section size in bytes */
    uint32_t sh_link; /* Link to another section */
    uint32_t sh_info; /* Additional section information */
    uint64_t sh_addralign; /* Section alignment */
    uint64_t sh_entsize; /* Entry size if section holds table */
} Elf64_Shdr;
```

Структура заголовков сегментов:

```
typedef struct {
    uint32_t p_type; /* Segment type */
    uint32_t p_flags; /* Segment flags */
    uint64_t p_offset; /* Segment file offset */
    uint64_t p_vaddr; /* Segment virtual address */
    uint64_t p_paddr; /* Segment physical address */
    uint64_t p_filesz; /* Segment size in file */
    uint64_t p_memsz; /* Segment size in memory */
    uint64_t p_align; /* Segment alignment */
} Elf64_Phdr;
```

⁵ Tool Interface Standard (TIS) Executable and Linking Format (ELF) Specification [Электронный ресурс]. URL: <https://refspecs.linuxbase.org/elf/elf.pdf> (дата обр. 28.02.2021)

- Portable Executable, PE⁶ – формат исполняемых файлов, объектного кода и динамических библиотек, используемый в 32- и 64-разрядных версиях операционной системы Microsoft Windows.

Структура MS-DOS заголовка файла:

```
typedef struct _IMAGE_DOS_HEADER { // DOS .EXE header
    WORD    e_magic;                // Magic number
    WORD    e_cblp;                // Bytes on last page of file
    WORD    e_cp;                  // Pages in file
    WORD    e_crlc;               // Relocations
    WORD    e_cparhdr;            // Size of header in paragraphs
    WORD    e_minalloc;           // Minimum extra paragraphs needed
    WORD    e_maxalloc;           // Maximum extra paragraphs needed
    WORD    e_ss;                 // Initial (relative) SS value
    WORD    e_sp;                 // Initial SP value
    WORD    e_csum;               // Checksum
    WORD    e_ip;                 // Initial IP value
    WORD    e_cs;                 // Initial (relative) CS value
    WORD    e_lfarlc;             // File address of relocation table
    WORD    e_ovno;               // Overlay number
    WORD    e_res[4];             // Reserved words
    WORD    e_oemid;              // OEM identifier (for e_oeminfo)
    WORD    e_oeminfo;           // OEM information; e_oemid specific
    WORD    e_res2[10];          // Reserved words
    LONG    e_lfanew;            // File address of new exe header
} IMAGE_DOS_HEADER
```

Структура PE-заголовка файла:

```
typedef struct _IMAGE_FILE_HEADER {
    WORD    Machine;
    WORD    NumberOfSections;
    DWORD   TimeDateStamp;
    DWORD   PointerToSymbolTable;
    DWORD   NumberOfSymbols;
    WORD    SizeOfOptionalHeader;
    WORD    Characteristics;
} IMAGE_FILE_HEADER, *PIMAGE_FILE_HEADER;
```

Структура заголовков секций:

```
typedef struct _IMAGE_SECTION_HEADER {
    BYTE    Name[8];
    union {
        DWORD   PhysicalAddress;
        DWORD   VirtualSize;
    } Misc;
    DWORD   VirtualAddress;
    DWORD   SizeOfRawData;
    DWORD   PointerToRawData;
    DWORD   PointerToRelocations;
    DWORD   PointerToLinenumbers;
    WORD    NumberOfRelocations;
    WORD    NumberOfLinenumbers;
    DWORD   Characteristics;
} IMAGE_SECTION_HEADER, *PIMAGE_SECTION_HEADER;
```

Структура таблицы экспортов:

```
typedef struct _IMAGE_EXPORT_DIRECTORY {
    DWORD   Characteristics;
```

⁶ PE Format - Win32 apps | Microsoft Docs, [Электронный ресурс]. URL: <https://docs.microsoft.com/en-us/windows/win32/debug/pe-format> (дата обр. 28.02.2021)

```

DWORD   TimeDateStamp;
WORD    MajorVersion;
WORD    MinorVersion;
DWORD   Name;
DWORD   Base;
DWORD   NumberOfFunctions;
DWORD   NumberOfNames;
DWORD   AddressOfFunctions;
DWORD   AddressOfNames;
DWORD   AddressOfNameOrdinals;
} IMAGE_EXPORT_DIRECTORY, *PIMAGE_EXPORT_DIRECTORY;

```

Структура дескриптора импортов:

```

typedef struct _IMAGE_IMPORT_DESCRIPTOR {
    union {
        DWORD   Characteristics;
        DWORD   OriginalFirstThunk;
    };
    DWORD   TimeDateStamp;
    DWORD   ForwarderChain;
    DWORD   Name;
    DWORD   FirstThunk;
} IMAGE_IMPORT_DESCRIPTOR;

```

- Mach object, Mach-O⁷ – формат исполняемых и объектных файлов, динамических библиотек и дампов памяти, использующийся в операционных системах Apple Inc.

Структура заголовка файла:

```

struct mach_header {
    unsigned long   magic;           /* Mach magic number identifier */
    cpu_type_t      cpustype;       /* cpu specifier */
    cpu_subtype_t   cpusubtype;     /* machine specifier */
    unsigned long   filetype;       /* type of file */
    unsigned long   ncmds;          /* number of load commands */
    unsigned long   sizeofcmds;     /* size of all load commands */
    unsigned long   flags;          /* flags */
};

```

Структура команд:

```

struct load_command {
    uint32_t   cmd;           /* type of load command */
    uint32_t   cmdsize;      /* size of load command */
};

```

Структура секций:

```

struct section {
    char        sectname[16]; /* section's name */
    char        segname[16]; /* segment the section is in */
    unsigned long   addr;     /* section's memory address */
    unsigned long   size;     /* section's size in bytes */
    unsigned long   offset;   /* section's file offset */
    unsigned long   align;    /* section's alignment */
    unsigned long   reloff;   /* file offset of relocation entries */
    unsigned long   nreloc;   /* number of relocation entries */
    unsigned long   flags;    /* flags */
    unsigned long   reserved1; /* reserved */
    unsigned long   reserved2; /* reserved */
};

```

⁷ Apple Open Source – Loader.h [Электронный ресурс]. URL: <https://opensource.apple.com/source/cctools/cctools-921/include/mach-o/loader.h.auto.html> (дата обр. 28.02.2021)

```
};
```

Структура сегментов:

```
struct segment_command {
    unsigned long cmd;           /* LC_SEGMENT */
    unsigned long cmdsize;      /* include size of section structures */
    char segname[16];          /* segment's name */
    unsigned long vmaddr;       /* segment's memory address */
    unsigned long vmsize;       /* segment's memory size */
    unsigned long fileoff;      /* segment's file offset */
    unsigned long filesize;     /* amount to map from file */
    vm_prot_t maxprot;          /* maximum VM protection */
    vm_prot_t initprot;         /* initial VM protection */
    unsigned long nsects;       /* number of sections */
    unsigned long flags;        /* flags */
};
```

Встроенные библиотеки

1. `struct`⁸ – этот модуль позволяет выполнять преобразования между значениями Python и структурами C, представленными в виде байтовых объектов Python. Он может быть использован при обработке данных, хранящихся в двоичных файлах, или полученных из других источников, в том числе, сетевых соединений.

Описание задач

1. Парсинг заголовка двоичных файлов формата ELF
 - Задание: получить информацию о типе, разрядности, точке входа, смещении таблицы заголовков программы, смещении таблицы заголовков секций из полей заголовка файла ELF.
 - Параметры программы:
 - Входные данные:
 - двоичный файл формата ELF.
 - Выходные данные:
 - тип файла;
 - разрядность;
 - виртуальный адрес точки входа;
 - смещение таблицы заголовков сегментов;
 - смещение таблицы заголовков секций.
2. Парсинг таблицы заголовков сегментов двоичных файлов формата ELF
 - Задание: получить информацию об имени, типе, смещении (офсете) и виртуальном адресе, а также указать список секций, входящих в сегмент из таблицы заголовков сегментов двоичных файлов формата ELF.
 - Параметры программы:
 - Входные данные:
 - двоичный файл формата ELF.

⁸ `struct` — Interpret bytes as packed binary data, [Электронный ресурс]. URL: <https://docs.python.org/3/library/struct.html> (дата обр. 28.02.2021)

- Выходные данные:
 - список сегментов:
 - тип;
 - смещение (офсет);
 - виртуальный адрес;
 - список секций, входящих в данный сегмент.
3. Парсинг таблицы заголовков секций двоичных файлов формата ELF
- Задание: получить информацию об имени секции, ее типе, смещении (офсете) и виртуальном адресе из таблицы заголовков секций двоичных файлов формата ELF.
 - Параметры программы:
 - Входные данные:
 - двоичный файл формата ELF.
 - Выходные данные:
 - список секций:
 - имя секции;
 - тип;
 - смещение (офсет);
 - виртуальный адрес
4. Парсинг импортов и экспортов двоичных файлов формата ELF
- Задание: получить информацию об импортируемых и экспортируемых функциях двоичных файлов формата ELF.
 - Параметры программы:
 - Входные данные:
 - двоичный файл формата ELF.
 - Выходные данные:
 - список функций.
5. Парсинг заголовка двоичных файлов формата Mach-O
- Задание: получить информацию о типе, разрядности, количестве команд загрузчику и флагах полей заголовка файла Mach-O.
 - Параметры программы:
 - Входные данные:
 - двоичный файл формата Mach-O.
 - Выходные данные:
 - тип файла;
 - разрядность;
 - количество команд загрузчику;
 - список флагов.
6. Парсинг заголовков команды загрузки сегментов двоичных файлов формата Mach-O

- Задание: получить информацию об имени, смещении (офсете) и виртуальном адресе, а также указать список секций, входящих в сегмент из команды загрузки сегментов двоичных файлов формата Mach-O (LC_SEGMENT/LC_SEGMENT_64).
- Параметры программы:
- Входные данные:
 - двоичный файл формата Mach-O.
- Выходные данные:
 - список сегментов:
 - имя;
 - смещение (офсет);
 - виртуальный адрес;
 - список секций, входящих в данный сегмент.

7. Парсинг заголовков секций двоичных файлов формата Mach-O

- Задание: получить информацию об имени секции и имени сегмента, в который входит данная секция, типе секции, смещении (офсете) и виртуальном адресе из заголовков секций двоичных файлов формата Mach-O.
- Параметры программы:
- Входные данные:
 - двоичный файл формата Mach-O.
- Выходные данные:
 - список секций:
 - имя секции;
 - тип;
 - смещение (офсет);
 - виртуальный адрес

8. Парсинг импортов и экспортов двоичных файлов формата Mach-O

- Задание: получить информацию об импортируемых и экспортируемых функциях двоичных файлов формата Mach-O.
- Параметры программы:
- Входные данные:
 - двоичный файл формата Mach-O.
- Выходные данные:
 - список функций.

9. Парсинг заголовка двоичных файлов формата PE

- Задание: получить информацию о типе, о времени, когда был собран данный файл, количестве секций и атрибутах, специфичных для данного файла, из полей заголовка файла PE.
- Параметры программы:
- Входные данные:

- двоичный файл формата PE.
- Выходные данные:
 - тип файла;
 - дата;
 - количество секций;
 - атрибуты.

10.Парсинг таблицы заголовков секций двоичных файлов формата PE

- Задание: получить информацию об имени секции, смещении (офсете) и виртуальном адресе, атрибутах, специфичных для данной секции, из таблицы заголовков секций двоичных файлов формата PE.
- Параметры программы:
- Входные данные:
 - двоичный файл формата PE.
- Выходные данные:
 - имя секции;
 - смещение (офсет);
 - виртуальный адрес;
 - атрибуты.

11.Парсинг импортов и экспортов двоичных файлов формата PE

- Задание: получить информацию об импортируемых и экспортируемых функциях двоичных файлов формата PE.
- Параметры программы:
- Входные данные:
 - двоичный файл формата PE.
- Выходные данные:
 - список функций.

Варианты

| № варианта | Список задач | № варианта | Список задач |
|------------|-------------------|------------|-------------------|
| 1 | 1, 4, 5, 8, 9, 10 | 11 | 1, 3, 5, 8, 9, 11 |
| 2 | 1, 4, 5, 7, 9, 11 | 12 | 1, 2, 5, 8, 9, 10 |
| 3 | 1, 4, 5, 6, 9, 11 | 13 | 1, 3, 5, 6, 9, 10 |
| 4 | 1, 4, 5, 8, 9, 11 | 14 | 1, 2, 5, 6, 9, 11 |
| 5 | 1, 4, 5, 7, 9, 10 | 15 | 1, 2, 5, 6, 9, 10 |
| 6 | 1, 3, 5, 7, 9, 11 | 16 | 1, 2, 5, 8, 9, 11 |
| 7 | 1, 2, 5, 7, 9, 10 | 17 | 1, 2, 5, 7, 9, 11 |
| 8 | 1, 3, 5, 6, 9, 11 | 18 | 1, 4, 5, 6, 9, 10 |
| 9 | 1, 3, 5, 7, 9, 10 | 19 | 1, 4, 5, 8, 9, 10 |
| 10 | 1, 3, 5, 8, 9, 10 | 20 | 1, 4, 5, 7, 9, 11 |

Практическая работа №4 «Реализация системы удаленного выполнения команд с использованием шифрования»

Цель работы: знакомство с криптографическими примитивами и получение навыков обмена данными между процессами с помощью сокетов.

Криптографические примитивы

– алгоритм Диффи-Хеллмана [5]:

криптографический протокол, позволяющий двум и более сторонам получить общий секретный ключ, используя не защищенный от прослушивания канал связи. Полученный ключ используется для шифрования дальнейшего обмена с помощью алгоритмов симметричного шифрования. Алгоритм состоит из следующих этапов:

1. Обоим абонентам известны некоторые числа p и g .
2. Оба абонента генерируют большие случайные числа a и b .
3. Первый абонент вычисляет остаток от деления $A = g^a \bmod p$ и пересылает его второму абоненту.
4. Второй абонент вычисляет остаток от деления $B = g^b \bmod p$ и пересылает его первому абоненту.
5. Первый абонент вычисляет ключ $K_A = B^a \bmod p = g^{ba} \bmod p$
6. Второй абонент вычисляет ключ $K_B = A^b \bmod p = g^{ab} \bmod p$
7. Нетрудно заметить, что первый и второй абонент получили один и тот же секретный ключ $K = K_A = K_B$

Реализация протокола Диффи-Хеллмана на языке Python с помощью библиотеки `cryptography` приведена в приложении 1.

– функция формирования ключа PBKDF2 [6]:

стандарт формирования ключа на основе пароля. Опции алгоритма:

- PRF — псевдослучайная функция, с выходом длины $hLen$;
- P — мастер-пароль;
- S — соль (salt);
- c — количество итераций, положительное целое число;
- $dkLen$ — желаемая длина ключа (не более $(2^{32} - 1) \times hLen$);
- Выходной параметр: DK — сгенерированный ключ длины $dkLen$.

Ход вычислений:

1. l — количество блоков длины $hLen$ в ключе (округление вверх), r — количество байт в последнем блоке:
 $l = \lceil dkLen/hLen \rceil$,
 $r = dkLen - (l - 1) \times hLen$.
2. Для каждого блока применить функцию F с параметрами P — мастер пароль, S — соль, c — количество итераций, и номером блока:
 $T_1 = F(P, S, c, 1)$,

$$T_2 = F(P, S, c, 2),$$

...

$$T_l = F(P, S, c, l),$$

F определена как операция *xor* (\oplus) над первыми c итерациями функции PRF, примененной к паролю P и объединению соли S и номеру блока, записанному как 4-байтовое целое с первым msb байтом.

$$F(P, S, c, i) = U_1 \oplus U_2 \oplus \dots \oplus U_c,$$

$$U_1 = PRF(P, S || INT(i)),$$

$$U_2 = PRF(P, U_1),$$

...

$$U_c = PRF(P, U_{c-1}).$$

3. Объединение полученных блоков составляет ключ DK . От последнего блока берется r байт.

$$DK = T_1 || T_2 || \dots || T_l < 0 \dots r - 1 >.$$

– алгоритм AES [7]:

симметричный алгоритм блочного шифрования (размер блока 128 бит, ключ 128/192/256 бит). Предварительно входные данные разбиваются на блоки по 16 байт, если полный размер не кратен 16 байтам, то данные дополняется до размера, кратного 16 байтам. Блоки представляются в виде матрицы 4x4 — state. Алгоритм состоит из следующих шагов:

1. Расширение ключа - KeyExpansion;
2. Начальный раунд - сложение state с основным ключом;
3. Девять раундов шифрования, каждый из которых состоит из преобразований:
 - SubBytes (замена байтов state по таблице S-box);
 - ShiftRows (циклический сдвиг строк state);
 - MixColumns (умножения каждого столбца state на фиксированную матрицу);
 - AddRoundKey (раундовый ключ поэлементно добавляется к state с помощью поразрядного XOR).
4. Финальный раунд, состоящий из преобразований:
 - SubBytes;
 - ShiftRows;
 - AddRoundKey.

Реализация алгоритма AES на языке Python с помощью библиотеки cryptography приведена в приложении 2.

– алгоритм Camellia [8]:

структура алгоритма основана на классической цепи Фейстеля с предварительным и финальным забеливанием. Цикловая функция

использует нелинейное преобразование (S-блоки), блок линейного рассеивания каждые 16 циклов (побайтовая операция XOR) и байтовую перестановку. В зависимости от длины ключа имеет 18 циклов (128-разрядный ключ), либо 24 цикла (192- и 256-разрядный ключ).

– алгоритм хэширования *SHA256* [9]:

Хеш-функции семейства SHA-2 построены на основе структуры Меркла — Дамгора. Исходное сообщение после дополнения разбивается на блоки, каждый блок — на 16 слов. Алгоритм пропускает каждый блок сообщения через цикл с 64 или 80 итерациями (раундами). На каждой итерации 2 слова преобразуются, функцию преобразования задают остальные слова. Результаты обработки каждого блока складываются, сумма является значением хеш-функции. Тем не менее, инициализация внутреннего состояния производится результатом обработки предыдущего блока. Поэтому независимо обрабатывать блоки и складывать результаты нельзя.

Встроенные библиотеки

1. `socket`⁹ – этот модуль обеспечивает доступ к интерфейсу сокета BSD. Он доступен во всех современных системах Unix, Windows, MacOS. Он включает в себя функции создания объекта сокета, который и обрабатывает канал данных, а также функции, связанные с сетевыми задачами, такими как преобразование имени сервера в IP адрес и форматирование данных для отправки по сети.

Сторонние библиотеки

2. `scapy`¹⁰ – это модуль, предназначенный для создания и манипулирования сетевыми пакетами, обеспечивающий поддержку большого количества сетевых протоколов.
3. `requests`¹¹ – это модуль, предназначенный для создания и отправки HTTP/1.1 запросов.
4. `cryptography`¹² – это модуль, предназначенный для работы с криптографическими алгоритмами и примитивами, включая функции формирования ключей, симметричные шифры, функции формирования имитовставки и другие.

⁹ `socket` — Low-level networking interface, [Электронный ресурс]. URL: <https://docs.python.org/3/library/socket.html> (дата обр. 28.02.2021)

¹⁰ Scapy — Packet crafting for Python2 and Python3, [Электронный ресурс]. URL: <https://scapy.net/> (дата обр. 28.02.2021)

¹¹ Requests: HTTP for Humans™, [Электронный ресурс]. URL: <https://docs.python-requests.org/en/master/> (дата обр. 28.02.2021)

¹² GitHub - pyca/cryptography: cryptography is a package designed to expose cryptographic primitives and recipes to Python developers, [Электронный ресурс]. URL: <https://github.com/pyca/cryptography> (дата обр. 28.02.2021)

Описание задачи

Необходимо реализовать систему удаленного выполнения команд, которая состоит из двух частей:

1. сервер, на котором удаленно выполняются команды
2. клиент, который отправляет команды на сервер и получает ответы

Для общения между сервером и клиентом используется интерфейс сокетов. Трафик между сервером и клиентом шифрованный. Также используются коды аутентификации (HMAC) для проверки целостности сообщения

Сообщение представляет собой конкатенацию следующих бинарных строк:

```
timestamp || IV || cipher_text || HMAC
```

где `timestamp` – временная метка создания сообщения; `IV` – вектор инициализации, который используется при шифровании в режиме CBC; `cipher_text` – зашифрованная команда; *HMAC* - код аутентификации сообщения, применяется к конкатенации следующих бинарных строк:
`timestamp || IV || cipher_text`

Для создания сеансового ключа предлагается использовать протокол Диффи-Хеллмана. Для дальнейшего преобразования длинного сеансового ключа в ключ, который будет использоваться для шифрования и генерации HMAC, предлагается использовать KDF (функцию формирования ключа).

Передаваемые команды имеют следующий вид:

```
{"command_number": <command_number>, "command_body": <command_body>}
```

где `<command_number>` – номер команды, которая будет выполнена; `<command_body>` – дополнительная информация, которая относится к команде. Например, различные флаги для команды. Этот параметр опционален (*не требуется для некоторых команд*).

Список команд, которые должны быть реализованы:

1. Выполнить произвольную `cmd`-команду.
2. Скачать файл.
3. Осуществить поиск файла:
 - 3.1. По названию.
 - 3.2. Содержащего определенный текст.
 - 3.3. По типу файла.
4. Получить список текущих процессов.
5. Сниффить трафик с интерфейса в течение определенного времени (`scapy`):
 - 5.1. Фильтрация по протоколу.
 - 5.2. Максимальное число пакетов.
6. Осуществить запрос на определенный `url`:
 - 6.1. Тип запроса.
 - 6.2. Получить куки.

- 6.3. Получить тело ответа.
7. Получить снимок экрана (изображение рекомендуется закодировать с помощью base64 перед отправкой).
 8. Взаимодействие с кейлоггером (информация сохраняется в файл):
 - 8.1. Запустить кейлоггер.
 - 8.2. Остановить кейлоггер.
 9. Cookie-stealer:
 - 9.1. Chrome
 - 9.2. Edge
 10. Password-stealer:
 - 10.1. Chrome
 - 10.2. Edge
 11. Получение истории браузера:
 - 11.1. Chrome
 - 11.2. Edge

Варианты

| № варианта | Протокол | KDF | Алгоритм шифрования | Паддинг | Хэш-функция |
|------------|----------|------------|---------------------|----------|-------------|
| 1 | TCP | PBKDF2HMAC | AES | PKCS7 | SHA3 256 |
| 2 | UDP | HKDF | Camellia | PKCS7 | Blake2b |
| 3 | TCP | PBKDF2HMAC | AES | PKCS7 | SHA256 |
| 4 | UDP | X963KDF | AES | PKCS7 | SHA256 |
| 5 | TCP | PBKDF2HMAC | ChaCha20 | PKCS7 | Blake2b |
| 6 | UDP | HKDF | ChaCha20 | ANSIX923 | SHA3 256 |
| 7 | TCP | PBKDF2HMAC | ChaCha20 | ANSIX923 | SHA256 |
| 8 | UDP | KBKDFHMAC | ChaCha20 | PKCS7 | SHA256 |
| 9 | TCP | X963KDF | ChaCha20 | PKCS7 | SHA3 256 |
| 10 | UDP | HKDF | AES | ANSIX923 | SHA256 |
| 11 | TCP | KBKDFHMAC | AES | ANSIX923 | Blake2b |
| 12 | UDP | HKDF | ChaCha20 | PKCS7 | SHA3 256 |
| 13 | TCP | Scrypt | ChaCha20 | ANSIX923 | SM3 |
| 14 | UDP | PBKDF2HMAC | Camellia | ANSIX923 | Blake2b |
| 15 | TCP | Scrypt | ChaCha20 | ANSIX923 | SHA256 |
| 16 | UDP | X963KDF | ChaCha20 | ANSIX923 | SHA3 256 |
| 17 | TCP | KBKDFHMAC | ChaCha20 | ANSIX923 | SHA3 256 |
| 18 | UDP | KBKDFHMAC | AES | ANSIX923 | SHA3 256 |
| 19 | TCP | PBKDF2HMAC | ChaCha20 | ANSIX923 | SM3 |
| 20 | UDP | HKDF | Camellia | ANSIX923 | SHA3 256 |

Список литературы

1. Лутц М. Изучаем Python. В 2-х т // Москва: Вильямс. – 2019.
2. Доусон М. Программируем на Python // СПб.: Питер. – 2014.
3. Seitz J. Gray Hat Python: Python Programming for Hackers and Reverse Engineers. // No Starch Press. – 2009.
4. Seitz J. Black Hat Python: Python Programming for Hackers and Pentesters. // No Starch Press. – 2014
5. Diffie W., Hellman M. New directions in cryptography //IEEE transactions on Information Theory. – 1976. – Т. 22. – №. 6. – С. 644-654.
6. Kaliski B. Rfc2898: Pkcs# 5: Password-based cryptography specification version 2.0. – 2000.
7. Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J. and T. Tokita, "Specification of Camellia --- a 128-bit Block Cipher", [Электронный ресурс]. URL: <http://info.isl.ntt.co.jp/camellia/> (дата обр. 28.02.2021)
8. Nechvatal J. et al. Report on the development of the Advanced Encryption Standard (AES) //Journal of Research of the National Institute of Standards and Technology. – 2001. – Т. 106. – №. 3. – С. 511.
9. Standard S. N. FIPS Pub 180-4 //National Institute of Standards and Technology. – 2015. – Т. 17. – С. 15.
10. Дауни, А. Б. Изучение сложных систем с помощью Python / А. Б. Дауни ; перевод с английского Д. А. Беликова. — Москва : ДМК Пресс, 2019. — 160 с. — ISBN 978-5-97060-712-1. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/131701> (дата обращения: 02.04.2021). — Режим доступа: для авториз. Пользователей.
11. Маккинни, У. Python и анализ данных / У. Маккинни ; перевод с английского А. А. Слинкина. — 2-ое изд., испр. и доп. — Москва : ДМК Пресс, 2020. — 540 с. — ISBN 978-5-97060-590-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/131721> (дата обращения: 05.04.2021). — Режим доступа: для авториз. Пользователей.
12. Бонцанини, М. Анализ социальных медиа на Python. Извлекайте и анализируйте данные из всех уголков социальной паутины на Python / М. Бонцанини ; перевод с английского А. В. Логунова. — Москва : ДМК Пресс, 2018. — 288 с. — ISBN 978-5-97060-574-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/108129> (дата обращения: 05.04.2021). — Режим доступа: для авториз. Пользователей.
13. Лучано, Р. Python. К вершинам мастерства / Р. Лучано ; перевод с английского А. А. Слинкин. — Москва : ДМК Пресс, 2016. — 768 с. — ISBN 978-5-97060-384-0. — Текст : электронный // Лань :

электронно-библиотечная система. — URL:
<https://e.lanbook.com/book/93273> (дата обращения: 05.04.2021). —
Режим доступа: для авториз. Пользователей.

14. Златопольский, Д. М. Основы программирования на языке Python / Д. М. Златопольский. — 2-ое изд., испр. и доп. — Москва : ДМК Пресс, 2018. — 396 с. — ISBN 978-5-97060-641-4. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/131683> (дата обращения: 05.04.2021). — Режим доступа: для авториз. пользователей.

Приложение 1. Реализация протокола Диффи-Хеллмана

```
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import dh
from cryptography.hazmat.primitives.kdf.hkdf import HKDF

# Создадим объект для генерации ключей
parameters = dh.generate_parameters(generator=2, key_size=2048)
# Получим закрытый ключ
private_key = parameters.generate_private_key()
public_key = private_key.public_key()
# Получим параметры, которые были сгенерированы
dh_numbers = parameters.parameter_numbers()

# Передадим параметры другой стороне и сгенерируем объект для генерации
ключей
peer_parameters = dh.DHParameterNumbers(dh_numbers.p,
dh_numbers.g).parameters()
# Сгенерируем закрытый ключ на другой стороне
peer_private_key = peer_parameters.generate_private_key()
# Получим публичный ключ другой стороны
peer_public_key = peer_private_key.public_key()

# На основе публичного ключа другой стороны создаем общий ключ
shared_key = private_key.exchange(peer_public_key)
# Аналогично получаем публичный ключ на другой стороне
peer_shared_key = peer_private_key.exchange(public_key)

# Преобразуем полученный общий ключ в ключ длиной 256-бит с помощью KDF
# KDF – функция, формирующая один или несколько секретных ключей на основе
секретного значения
# (главный ключ, пароль или парольная фраза) с помощью псевдослучайной
функции.
# Создаем объект HKDF, с помощью которого создадим ключ для симметричного
шифрования
hkdf_obj = HKDF(
    algorithm=hashes.SHA256(),
    length=32,
    salt=None,
    info=b"",
)
peer_hkdf_obj = HKDF(
    algorithm=hashes.SHA256(),
    length=32,
    salt=None,
    info=b"",
)
# Получим ключ длиной 256-бит
derived_key = hkdf_obj.derive(shared_key)
# Получим ключ длиной 256-бит на другой стороны
peer_derived_key = peer_hkdf_obj.derive(peer_shared_key)
# Сравним полученные ключи
print(f"Are keys equal: {derived_key == peer_derived_key}")
```

Приложение 2. Реализация алгоритма AES

```
import os
from cryptography.hazmat.primitives import padding
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms,
modes

def check_cipher_suite(cipher_suite, padding_algorithm):
    # Получим объекты для шифрования и-padding
    encryptor = cipher_suite.encryptor()
    padder = padding_algorithm.padder()

    # Текст, который будем шифровать
    plain_text = b"ITMO"
    # Сделаем padding текста
    padded_plain_text = padder.update(plain_text) + padder.finalize()
    print(f"Padded text: {padded_plain_text}")
    # Зашифруем текст
    cipher_text = encryptor.update(padded_plain_text) +
encryptor.finalize()
    print(f"Cipher text: {cipher_text}")

    # Получим объекты для дешифрования и анpadding
    decryptor = cipher_suite.decryptor()
    unpadder = padding_algorithm.unpadder()

    # Получим расшифрованный текст, к которому применен padding
    padded_plain_text = decryptor.update(cipher_text) +
decryptor.finalize()
    print(f"Decrypted text: {padded_plain_text}")
    # Получим текст, который был зашифрован
    plain_text = unpadder.update(padded_plain_text) + unpadder.finalize()
    print(f"Unpadded text: {plain_text}")

# Сгенерируем ключ и вектор инициализации
key = os.urandom(16) # 128-битовый ключ
iv = os.urandom(16) # вектор инициализации
# Создадим объект с набором инструментов для работы с AES в режиме CBC
cipher_suite = Cipher(algorithms.AES(key), modes.CBC(iv))
# Для padding будем использовать PKCS7. Размер блока шифрования – 128 бит
(AES)
check_cipher_suite(cipher_suite, padding.PKCS7(128))
print()

key = os.urandom(16) # 128-битовый ключ
# Создадим объект с набором инструментов для работы с AES в режиме ECB
cipher_suite = Cipher(algorithms.AES(key), modes.ECB())
# Для padding будем использовать PKCS7. Размер блока шифрования – 128 бит
(AES)
check_cipher_suite(cipher_suite, padding.PKCS7(128))
print()

key = os.urandom(16) # 128-битовый ключ
# Создадим объект с набором инструментов для работы с AES в режиме ECB
cipher_suite = Cipher(algorithms.AES(key), modes.ECB())
# Для padding будем использовать ANSIX923. Размер блока шифрования – 128
бит (AES)
check_cipher_suite(cipher_suite, padding.ANSIX923(128))
print()
```

```
key = os.urandom(24) # 192-битовый ключ
# Создадим объект с набором инструментов для работы с Camellia в режиме ECB
cipher_suite = Cipher(algorithms.Camellia(key), modes.ECB())
# Для паддинга будем использовать PKCS7. Размер блока шифрования – 128 бит
(Camellia)
check_cipher_suite(cipher_suite, padding.PKCS7(128))
print()
```

Леонов Федор Вадимович
Челпанов Антон Дмитриевич
Югансон Андрей Николаевич

**Программирование на языке Python для решения задач
информационной безопасности. Методические указания по
выполнению практических работ.**

Учебно-методическое пособие

В авторской редакции

Редакционно-издательский отдел Университета ИТМО

Зав. РИО

Н.Ф. Гусарова

Подписано к печати

Заказ №

Тираж

Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49, литер А