ВВЕДЕНИЕ

Цель работы – реализовать незаметный сбор данных о пользователе и его устройстве при обновлении программы и про посещении пользователя веб страницы.

Для достижения поставленной цели необходимо решить следующие задачи:

- создать программу, которая будет выполнять «обновления»;
- реализовать скрипт, который будет собирать данные о пользователе и помещать их в файл;
- реализовать кодирование этого файла, чтобы нельзя было узнать его содержимое;
- реализовать защиту файлов от изменения;
- реализовать отключение скрипта при вводе пароля;
- проделать те же пункты для веб странички;
- проверить работоспособность данной программы;
- описать ход выполнения работы в отчете.

1 СБОР ИНФОРМАЦИИ ЛОКАЛЬНО С УСТРОЙСТВА

1.1 Задание

1. Создать текстовый документ (sys.tat), в котором будет содержаться «Системная информация».
2. Написать программу-инсталлятор sys_doc.exe (я запускал программу sys_doc.exe.py) для этого документа, которая под видом установки обновления (с отображением строки прогресса обновления) к программе):

- запрашивает у пользователя папку для копирования «Системной информации»;
- записывает в папку файл с исполняемым кодом программы secur.exe, защищающей sys.tat;
- собирает (возможную) информацию о компьютере;
- кодирует эту информацию и записывает в файл new_file_info.txt;
- подписывает её личным ключом пользователя программы и записывает подпись, например Фамилию;
- запускает secur.exe для защиты sys.tat от несанкционированного доступа;
- прописывает запуск программы secur.exe при выполнении функции Open для sys.tat, чтобы защита срабатывала и после перезагрузки ОС.

3. В саму программу защиты secur.exe включить следующий функционал:

- запрос у пользователя информации об имени раздела реестра с электронной цифровой подписью (фамилией студента);
- разрешение или запрет просмотра «Системной информации» в файле sys.tat в зависимости от правильности указания ключа.

4. При неудачной проверке работа защищаемой программы должна прекращаться с выдачей соответствующего сообщения.

1.2 Ход работы

Возьмем файлы из лабораторной 1, которые будут защищать файл sys.tat от изменения – block.sh и unblock.sh и создаем основной файл sys_doc.exe.py, который будет создавать вид обновления (рисунок 1).

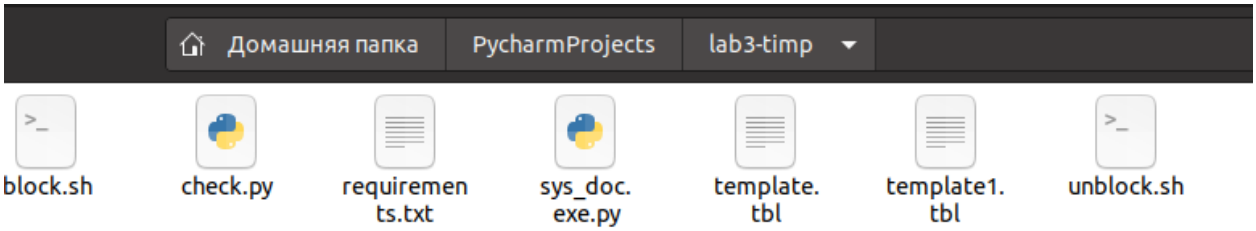

Рисунок 1 – создадим файлы

Запускаем программу sys_doc.exe и указываем обновление программы Notepad и создание новой директории Notepad (рисунок 2).

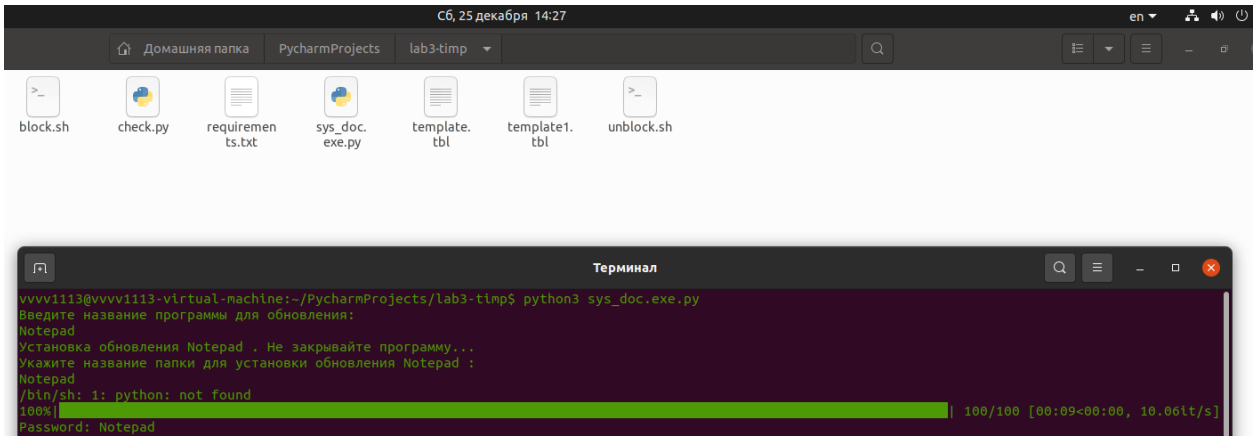

Рисунок 2 – Запуск выполнения «обновления»

Вводим пароль, например, тот же Notepad, после чего у нас создается файл sys.tat (рисунок 3).

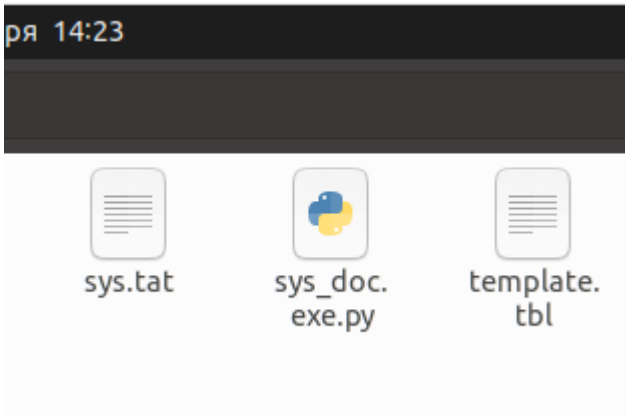

Рисунок 3 – Создаются новые файлы

«Открываем» файл командой open и видим, что теперь файл sys.tat запрещен для чтения (рисунок 4).

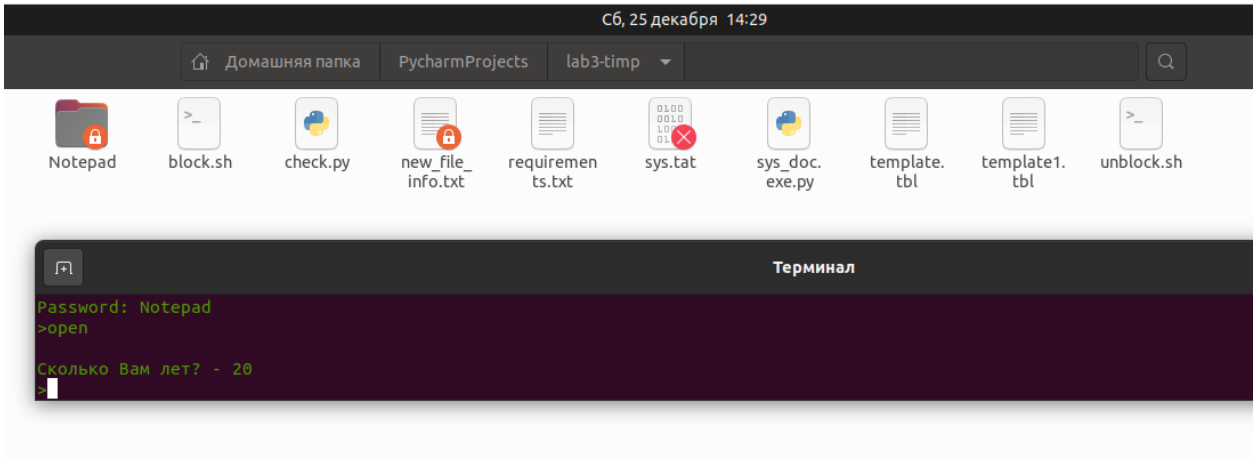

Рисунок 4 – Запуск выполнения локальной программы

Также здесь создается новый файл new_file_info.txt

Пробуем открыть sys.tat (рисунок 5) и видим, что это невозможно.

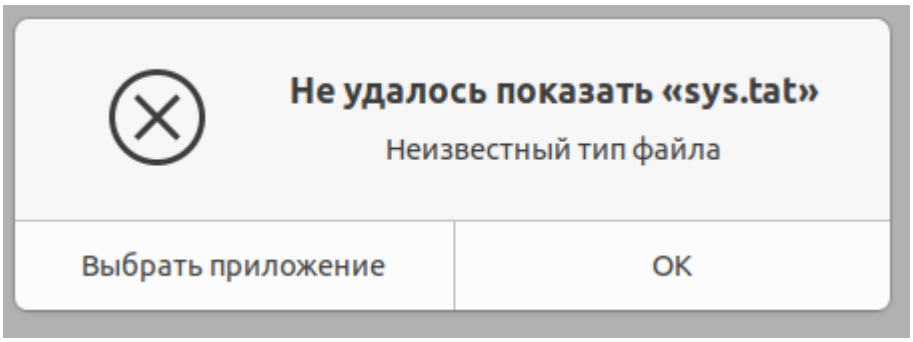

Рисунок 5 – Попытка прочитать файл sys.tat

Изменим права доступа к файлам при помощи кода, который был использован для лабораторной 1. Для этого введем локальную команду stop, а затем пароль – Notepad. Видим, что теперь файл доступен для чтения (рисунок 6).

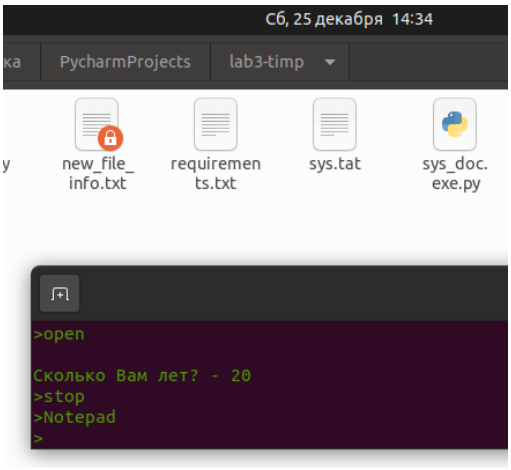

Рисунок 6 – Изменение прав доступа к файлу sys.tat

Открываем файл и видим, что в нем собрана информация об устройстве (рисунок 7).

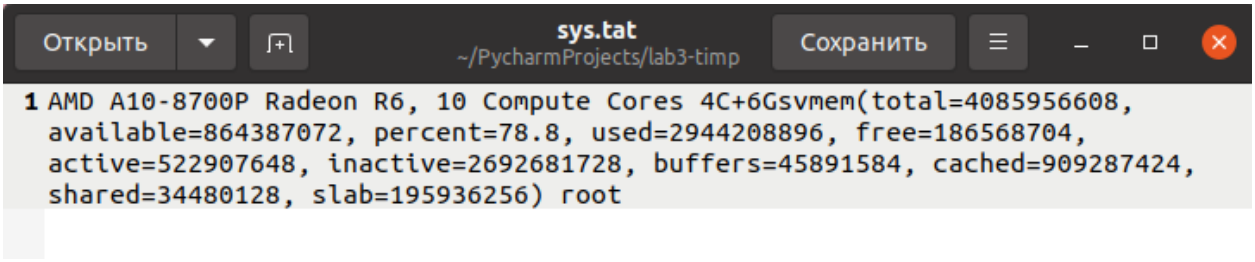

Рисунок 7 – Содержимое файла sys.tat

Откроем файл new_tetx_info.txt, видим что он был закодирован (рисунок 8).

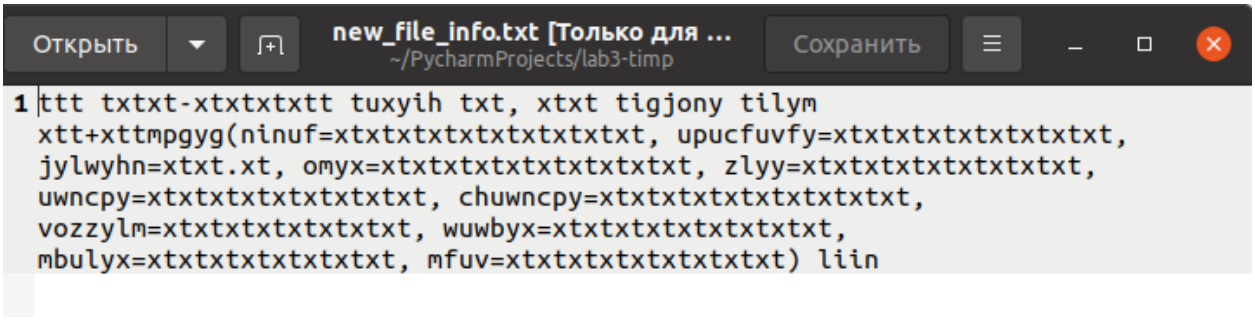

Рисунок 8 – Содержимое файла new_tetx_info.txt

При вводе команды env, программа вывод подписанный ключ shepelev (рисунок 9).

Рисунок 9 – Подписной ключ

При вводе команды def видим, что полученные файлы снова защищаются от изменения и изменяются из права (рисунок 10).

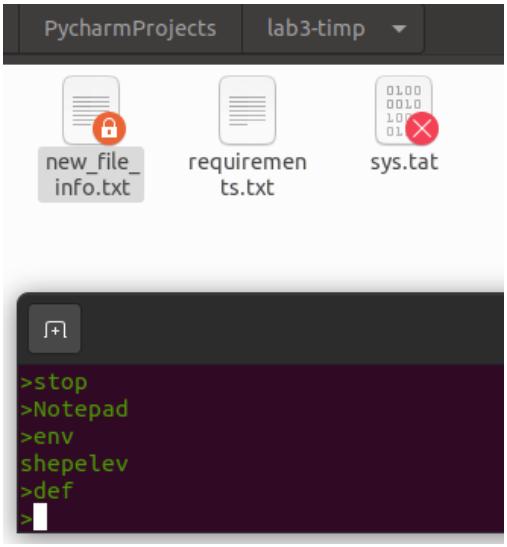

Рисунок 10 – Защита файлов от изменения и чтения

Листинг 1 – изменение настроек защиты программы внутри ее выполнения.

```
password = input("Password: ")
    cmd = input('>')
    while cmd != 'close':
        if cmd == 'open':
            files = crypt_files(password)
            os.chmod("sys.tat", 0o333)
            rc = call("./block.sh", shell=True)
        elif cmd == 'def':
            os.chmod("sys.tat", 0o333)
            rc = call("./block.sh", shell=True)
        elif cmd == 'stop':
            stop = True
            rc = call("./unblock.sh", shell=True)
        elif cmd == 'env':
            print(os.environ['shepelev'])
        elif cmd == 'close':
            rc = call("./unblock.sh", shell=True)
        elif cmd == password:
            os.chmod("sys.tat", 0o777)
        cmd = input('>')
```

Программа работает, она собирает информацию о пользователе, создает новую папку и купирует туда исполняемый код на python. При вводе защиты она ее кодирует и изменяет права доступа. Все функциональности выполняются.

2 СБОР ИНФОРМАЦИИ УДАЛЕННО

2.1 Задание

1. Создать скрипт, который удалённо и незаметно для пользователя (пользователь открывает какую-нибудь веб-страничку от создателя скрипта) собирает информацию о нём, его компьютере и системе и записывает её на какой-либо локальный сетевой диск (доступный создателю скрипта) в папку с именем IP или Mac-адреса пользовательской машины.
2. Продумать доступ к этой информации.
3. Протестировать на 3-5 клиентах и получить статистику о них.

2.2 Ход работы

Создадим проект на flask.

Создадим основную страницу index.html. На странице реализована анимация, код которой написан на css (листинг 2 – находится в конце этого раздела отчета).

При запуске веб страницы мы видим эту анимацию (рисуно 11).

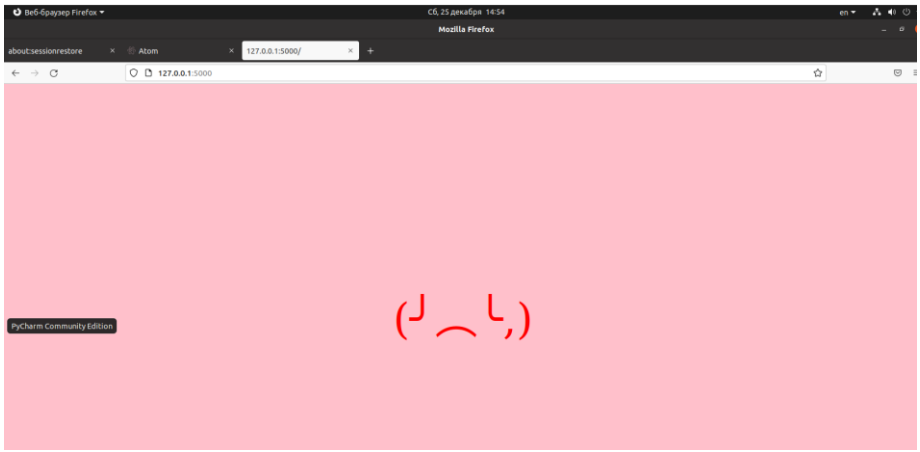

Рисунок 11 – Анимация на основной странице

При запуске самой страницы реализован скрипт, который незаметно собирает информацию о компьютере, его OS, версии OS, модель компьютера, память ядра (рисунок 12).

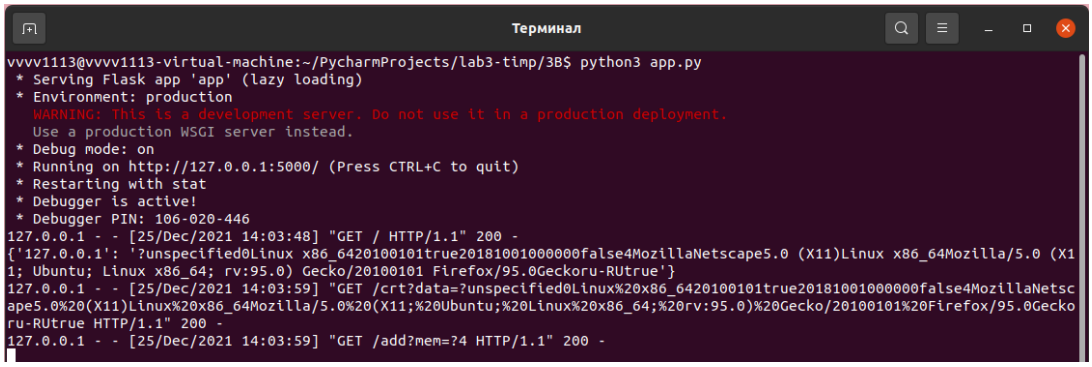

Рисунок 12 – Вывод собранной информации на консоль

Вся информация выводится в локальный файл (листинг 4), который хранится на моем ПК. Я протестировал работу программы на своих знакомых. Я дал им попользоваться ссылкой на данную страницу, которую я разместил в общем доступе и собрал информацию (листинг 3) об их устройствах (рисунок 13).

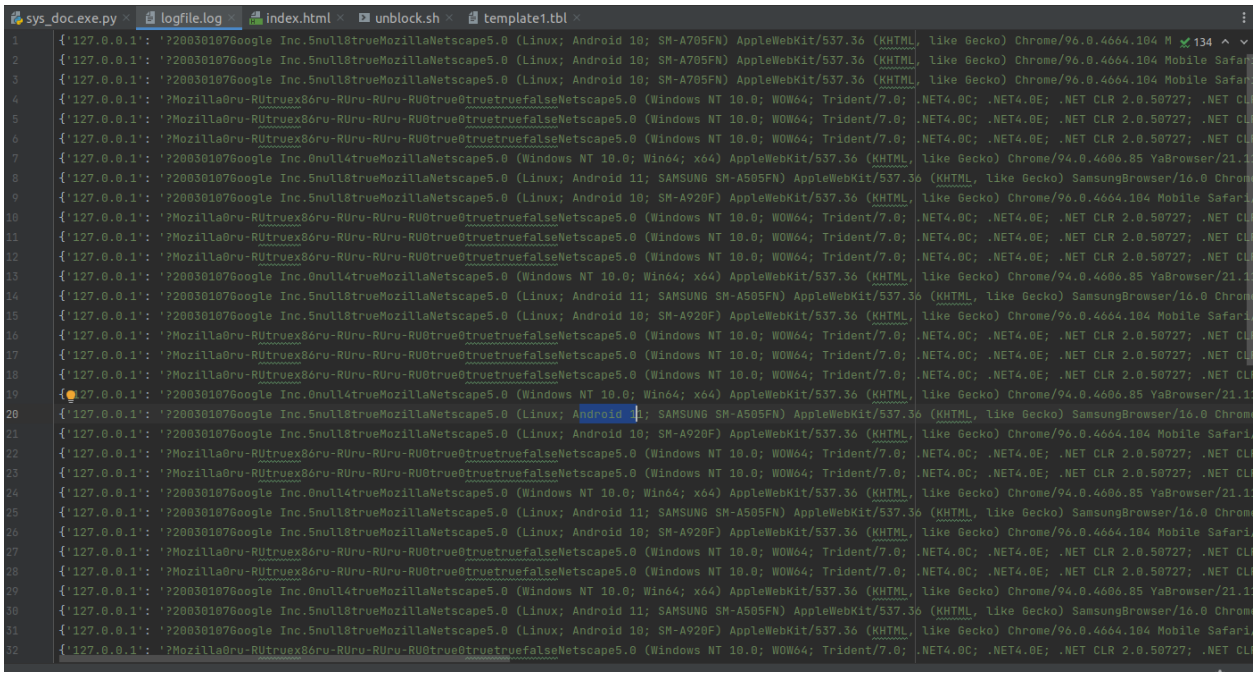

Рисунок 13 – Собранная информация об устройствах

Затем я проделал защиту данного файла для чтения и изменения подобно предыдущему пункту данной лабораторной работы.

Листинг 2 – CSS-код основной страницы

```
.load:after {
  animation: changeContent 66.6s linear infinite;
  display: block;
  content: "";
  font-size: 80px;
}


@keyframes Content {
   1.5% { content: "10 секунд..."; }
  3% { content: "9"; }
  4.5% { content: "8"; }
  6% { content: "7"; }
  7.5% { content: "6"; }
```

```
9% { content: "5"; }
10.5% { content: "4"; }
12% { content: "3"; }
13.5% { content: "2"; }
15% { content: "1"; }
16.5% { content: "Загрузка..."; }
17% { content: "▉▒▒▒▒▒▒▒▒▒ 10%"; }
17.5% { content: "▉▉▒▒▒▒▒▒▒▒ 20%"; }
18.5% { content: "▉▉▉▒▒▒▒▒▒▒ 30%"; }
18.75% { content: "▉▉▉▉▒▒▒▒▒▒ 40%"; }
19.25% { content: "▉▉▉▉▉▒▒▒▒▒ 50%"; }
19.5% { content: "▉▉▉▉▉▒▒▒▒▒ 60%"; }
20.5% { content: "▉▉▉▉▉▉▒▒▒▒ 70%"; }
21% { content: "▉▉▉▉▉▉▉▒▒▒ 80%"; }
21.5% { content: "▉▉▉▉▉▉▉▉▒▒ 90%"; }
22.5% { content: "▉▉▉▉▉▉▉▉▉ 99%"; }
23.5% { content: "♟"; }
23.75% { content: "♞"; }
24% { content: "♝"; }
24.25% { content: "♜"; }
24.5% { content: "♛"; }
24.75% { content: "♚"; }
25% { content: "♙"; }
25.25% { content: "♘"; }
25.5% { content: "♗"; }
25.75% { content: "♖"; }
26% { content: "♕"; }
26.25% { content: "♔"; }
26.5% { content: "♟"; }
26.75% { content: "♞"; }
27% { content: "♝"; }
27.25% { content: "♜"; }
```

```
  27.5% { content: "♛"; }
  27.75% { content: "♚"; }
  28% { content: "♟"; }
  28.5% { content: "♘"; }
  29.25% { content: "♗"; }
  30.25% { content: "♖"; }
  31.5% { content: "♕"; }
  33% { content: "♔"; }
  34.75% { content: "♞"; }
  36.75% { content: "♛"; }
  39% { content: "♔"; }
  41.5% { content: "♘"; }
  44.5% { content: "♝"; }
  49.5% { content: "Connection is lost!"; }
  56% { content: "(╯﹏╰,)"; }
}
```

Листинг 3 – скрипт, собирающий информацию, с комментариями

```
function CreateRequest()
{
    var Request = false;
    if (window.XMLHttpRequest)
    {
        Request = new XMLHttpRequest();
    }


    else if (window.ActiveXObject)
    {
        //Internet explorer
        try
        {
```

```
            Request = new ActiveXObject("Microsoft.XMLHTTP");
        }
        catch (CatchException)
        {
            Request = new ActiveXObject("Msxml2.XMLHTTP");
        }
    }
    if (!Request)
    {
        alert("Troubles");
    }
    return Request;
}
    function SendRequest(r_method, r_path, r_args)
{
    var Request = CreateRequest(); //request
    if (!Request) // checking of request
    {
        alert("Troubles");
        return;
    }
    if (r_method.toLowerCase() == "get" && r_args.length > 0)
//checking get-request

    Request.open(r_method, r_path, true); //create a connection
    if (r_method.toLowerCase() == "post") // If this is a post req
    {
        Request.setRequestHeader("Content-Type","application/x-
www-form-urlencoded; charset=utf-8"); // set the Header
        Request.send(r_args); // send req
    }
    else
    {
        Request.send(null); // if this is ger req, then i send a
null-req
```

```
    }
}
```

Листинг 4 – Перенаправление данных в локальный файл

```
(function(console){

    console.save = function(data, filename){

        if(!data) {
            console.error('Console.save: No data')
            return;
        }

        if(!filename) filename = 'console.json'

        if(typeof data === "object"){
            data = JSON.stringify(data, undefined, 4)
        }

        var blob = new Blob([data], {type: 'text/json'}),
            e    = document.createEvent('MouseEvents'),
            a    = document.createElement('a')

        a.download = filename
        a.href = window.URL.createObjectURL(blob)
        a.dataset.downloadurl   =     ['text/json',   a.download,
a.href].join(':')
        e.initMouseEvent('click', true, false, window, 0, 0, 0, 0,
0, false, false, false, false, 0, null)
        a.dispatchEvent(e)
    }
})(console)
```