МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информатики и прикладной математики

Кафедра вычислительных систем и программирования

КОНТРОЛЬНАЯ РАБОТА

по дисциплине:

«Программно-аппаратные средства защиты информации»

Вариант (тема) Лабораторная работа №2

Направление 10.03.01 «Информационная безопасность»

Направленность (профиль) «Безопасность компьютерных систем»

Студент Тесленко Марк Сергеевич

(Ф.И.О. полностью)

Группа ИБ-1502 Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(номер группы)

Проверил Федоров Дмитрий Юрьевич

(Фамилия И.О. преподавателя)

Должность Старший преподаватель

(уч. степень, уч. звание)

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дата:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись:

Санкт-Петербург

2017

[1. Постановка задачи 3](#_Toc492630887)

[2. Выполненная работа 4](#_Toc492630888)

[3. Выводы 7](#_Toc492630889)

# 1. Постановка задачи

Разработайте распределенную систему мониторинга удаленных хостов:Каждый хост (операционная система на выбор разработчика) содержит программу-агента, который собирает информацию о текущем состоянии системы, например, контроль запуска определенных служб (контролируемые службы выбираются на усмотрение разработчика, можно реализовать выбор службы для мониторинга через конфигурационный файл). Необходимо задействовать максимальные возможности Python

для работы с операционными системами.

На хосте производится сбор основных действий агента и результатов мониторинга (время, состояние и пр.). Через определенные интервалы времени агенты отправляют информацию на центральный сервер мониторинга. Сервер мониторинга опрашивает агентов, в ответ получает информацию о текущем состоянии системы. На сервере мониторинга производится логирование основных действий и результатов сбора

информации (IP-адрес хоста, время и пр.). Итоговый результат сбора информации представляется в виде таблицы или (желательно) графика.

При реализации системы необходимо задействовать возможности библиотек языка программирования Python (os, xmlrpclib и пр.). В качестве хранилища данных можно использовать текстовые файлы собственного формата, XML-формат, БД (MySQL, SQLite)..

# 2. Выполненная работа

* Файл server.py

import socket

import time

import pickle

import json

import string

HOST = '127.0.0.1'

PORT = 50009

def config\_connection(HOST,PORT):

s = socket.socket()

try:

s.bind((HOST, PORT))

except ConnectionError:

print("Ошибка соединения с сервером")

exit(2)

s.listen(5)

conn, addr = s.accept()

#print('Connected client',addr)

conn.send(byte\_processes)

print('Config files sended: ', processes)

conn.close()

try:

with open("conf.txt") as file:#получение из конф.файла имена нужных процессов

processes = [row.strip() for row in file]

except FileNotFoundError:

print("Ошибка открытия конфигурационного файла")

exit(1)

byte\_processes = pickle.dumps(processes, protocol=0)# перевод процессов в байт-код

config\_connection(HOST,PORT)

#вторая передача

while 1:

s = socket.socket()

try:

s.bind((HOST, PORT))

except ConnectionError:

print("Ошибка соединения с сервером")

exit(3)

s.listen(5)

conn, addr = s.accept()

#print('Connected client',addr)

while 1:

pid\_time\_pickled = conn.recv(4096)

if not pid\_time\_pickled:

break

else:

try:

pickleddata=pickle.loads(pid\_time\_pickled)

except Exception:

print("Конфигурационные данные повреждены / не получены в полном объеме")

exit(9)

print('Connected client',addr)

print("Название".ljust(15),"Время работы".rjust(15))

for key in pickleddata:

print (key.ljust(15),pickleddata[key].rjust(15))

print('\n')

conn.close()

with open ('monitor.txt','a') as f:#запись словаря с полученными данными в файл

for key in pickleddata:

f.write(key.ljust(15))

f.write(pickleddata[key].rjust(15))

f.write('\n')

* Файл client.py

import os, string, subprocess, re

import urllib.request

import urllib.parse

import socket

import pickle

import time

HOST = '127.0.0.1'

PORT = 50009

def getpid(name\_of\_process): #функция получения pid процесса,в аргументе имя

req="pidof "+name\_of\_process +"|awk '{print$1}'"

stdoutdata1 = subprocess.check\_output(req, shell=True)

stdoutdata1=str(stdoutdata1)

result1 = re.findall(r'\d', stdoutdata1)

mystr = ''.join(result1) #склеивание элементов списка result1 в строку для удобства

if len(mystr)==0:#dlina mystr 0 esli process ne zapushen

print("Приложение "+name\_of\_process+" не запущено")

return None

else:

return mystr

def process\_time(pid): #функция получения времени работы процесса,в аргументе pid

result = []

req1="ps -eo pid,etime | grep "+pid+" | awk 'NR == 1{print$2}'" #команда bash

stdoutdata = subprocess.check\_output(req1, shell=True)

stdoutdata=str(stdoutdata)

stdoutdata = stdoutdata.replace('b', '') #удаление

stdoutdata = stdoutdata.replace('\\n', '') #лишних

stdoutdata = stdoutdata.replace('\'', '') #символов

result.insert(0, stdoutdata)

return (result[0])

pid\_time = {}

s = socket.socket()

try: #проверка на ошибку соединения

s.connect((HOST, PORT))

except ConnectionError:

print("Ошибка соединения с сервером1")

exit(1)

try: #проверка целостность полученных данных

process\_names = s.recv(4096)

s.close()

print('Received[4]: ', pickle.loads(process\_names))

except Exception:

print("Конфигурационные данные повреждены / не получены в полном объеме")

exit(9)

process\_names = pickle.loads(process\_names)#передача в список названий полученных процессов

while 1:

for i in range(len(process\_names)): # в цикле перебираются полученные имена и выводится словарь {pid:time}

pid = getpid(process\_names[i])

if (pid == None): #если ф-ия вернула none, значит приложение не запущено

continue

pid\_time[process\_names[i]] = process\_time(pid)

print("Запущенные приложения:", end=' ')

print(pid\_time)

s = socket.socket()

try:

s.connect((HOST, PORT))

except ConnectionError:

print("Соединение с сервером потеряно")

exit(2)

pid\_time\_pickled = pickle.dumps(pid\_time, protocol=0)

s.send(pid\_time\_pickled)

s.close()

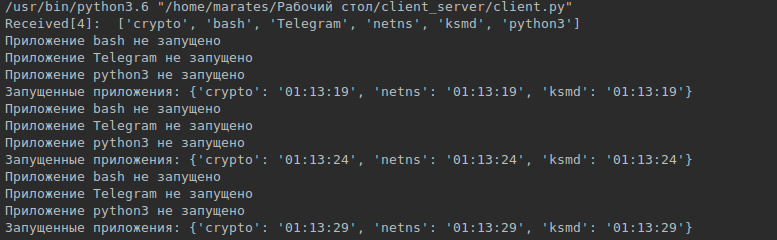
pid\_time={} #обнуление словаря

time.sleep(5) #пауза между отправкой на сервер данных о запущенных приложениях(в сек)

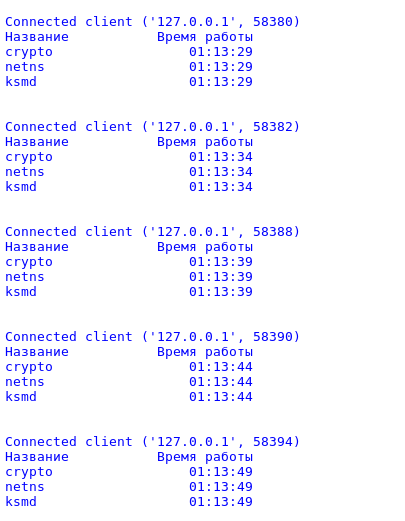
# в result[0] хранится время работы процесса

#ps -eo pid,etime | grep $PID | awk '{print $2}' вывести время работы процесса!

* Скриншот работы (client)



* Скриншот работы (server)



# 3. Выводы

В ходе выполнения лабораторной работы №2 у меня получилось создать распределенную систему мониторинга удаленных хостов на Python, которая запрашивает информацию у удаленных хостов о времени работы конкретных приложений и получает форматированный ответ.

Проделанная работа показалось мне очень интересной и творческой, я потратил на нее много времени и нервов.