Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Петрозаводский государственный университет
Институт математики и информационных технологий
Кафедра информатики и математического обеспечения

Отчет по дисциплине «Верификация ПО»

ПРОГРАММНАЯ СИСТЕМА СБОРА И АНАЛИЗА ДАННЫХ АКТИВНОСТИ ПОЛЬЗОВАТЕЛЕЙ БЕСПРОВОДНЫХ СЕТЕЙ

Выполнил:
магистрант 2 года группы 22608 А. В. Чувак

_____

Лектор:
к.ф-м.н., доцент К. А. Кулаков
Итоговая оценка: _____

_____

Петрозаводск
2016

Оглавление

1. Описание объекта тестирования

Объектом тестирования является консольное приложение для сбора и статистической обработки данных об активности пользователей беспроводной компьютерной сети. Приложение реализовано на языке Python с использованием сторонних библиотек openpyxl [1] (для работы с файлами MS Excel), jsonschema [2] (для валидации JSON-структур), manuf [3] (для установления производителя оборудования по MAC-адресу) и внешней (для приложения) утилиты Nmap [4] (для сбора данных). Приложение спроектировано и реализовано для функционирования в среде Windows 10.

Основные функциональности объекта тестирования, предназначенные для использования пользователем:

1. периодический опрос локальной беспроводной сети (в которой находится компьютер с запущенным приложением) с помощью протокола ARP используя утилиту Nmap;
2. автоматическое переключение между заданным списком сетей Wi-Fi (если опрос проводится в беспроводных локальных сетях) и опрос каждой из этих сетей;
3. формирование файлов MSExcel, содержащих таблицы с количеством активных пользователей в каждой из сетей за каждую из итераций опроса сети и с распределением устройств пользователей по производителям;

Дополнительные функциональности объекта тестирования (использующиеся самой системой для обеспечения работы основных функциональностей):

1. разбор аргументов командной строки, переданных приложению, и выполнение задач в соответствии с указанными параметрами;
2. формирование отчета о распределении активных в сети устройств между производителями оборудования (используя MAC-адреса устройств) за заданных временной период;

3. разбор XML-файлов отчетов, предоставляемых Nmap, формирование на их основе JSON файлов (для экономии дискового пространства) и сохранение этих файлов на диск;
4. автоматическое определение IP-адреса и маски подсети, к которой подключен запускающий программу компьютер;
5. разбор сохраненных JSON-файлов с целью дальнейшей совместной обработки данных из этих файлов;

2. Стратегия тестирования

2.1. Структура приложения

Приложение состоит из четырех модулей и одной внешней утилиты.

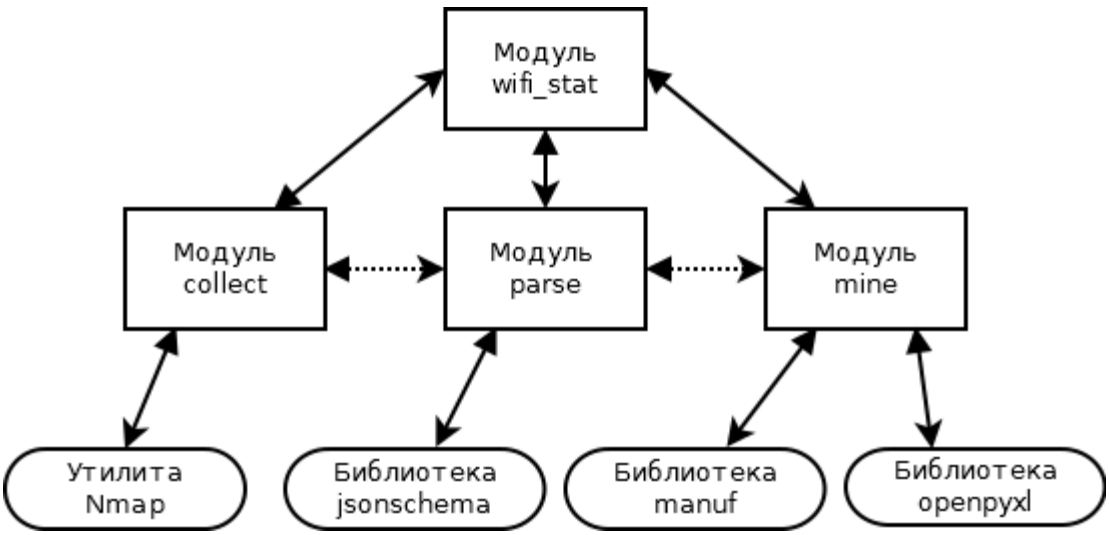

Рисунок 1 Структура приложения

Далее следует описание элементов приложения.

- Модуль *wifi_stat* является точкой входа в приложение. Он осуществляет разбор аргументов командной строки и инициирует выполнение задания на основе переданных аргументов, для чего вызывает функции других модулей.
- Модуль *collect* служит для взаимодействия с сетью и опроса сети. Содержит следующие функции:
  - *get_network_address*определяет IP-адрес и маску подсети, к которой подключен компьютер;

- *get_wlan_name* определяет SSID беспроводной сети, к которой подключен компьютер;
- *connect* производит переподключение сетевого адаптера компьютера к заданной по имени Wi-Fi сети;
- *parse_nmap_xml*производит разбор XML-вывода программы Nmap;
- *collect* инициирует периодический опрос сети с помощью утилиты Nmap (к которой осуществляется системный вызов); опрос осуществляется через заданные интервалы времени; перед каждым опросом осуществляется подключение к другой сети (если опрашивается беспроводная сеть), определение адреса новой сети, сохранение данных об итерации опроса сети на диск в виде JSON-файлов;

- Модуль *parse* осуществляет разбор и валидацию JSON-файлов, которые создаются после каждой итерации опроса сети. Модуль содержит следующие функции:
  - *parse_data* читает JSON-файлы и составляет из них ассоциативный массив следующей структуры:
    - имя сети
      - дата и время сбора данных
        - количество активных пользователей
        - адреса активных пользователей
        - дата и время начала сбора данных
        - дата и время окончания сбора данных
        - адрес подсети
        - имя беспроводной сети (SSID)
- Модуль *mine* служит для статистической обработки собранных данных. Он содержит следующие функции:

- *excel* формирует и записывает на диск файл MSExcel, содержащих таблицы (листы) с количеством активных пользователей в каждой из сетей за каждую из итераций опроса сети;
- *get_addresses* составляет множество MAC-адресов активных устройств
- *count_vendor_devices* составляет ассоциативный массив со сводкой о производителях оборудования среди активных пользователей;

2.2. Связи между элементами приложения

Взаимодействие между модулем *collect* и утилитой Nmap осуществляется с помощью системных вызовов, инициируемых с помощью стандартной библиотеки языка Python и XML-вывода программы Nmap.

Взаимодействие между модулями *collect* и *parse* осуществляется посредством JSON-файлов, которые формирует модуль *collect* и сохраняет на диск.

Взаимодействие между модулями *parse* и *mine* осуществляется через модуль *wifi_stat*: данные, полученные при разборе файлов в модуле *parse* передаются в качестве аргумента функций модуля *mine*.

Утилита Nmap вызывается из модуля collect; библиотека jsonschema используется в функции parse_data модуля parse; библиотека manuf используется в функции count_vendor_devices модуля mine; библиотека openpyxl используется в функции excel модуля mine.

2.3. Стратегия блочного тестирования

Для проведения блочного тестирования необходимо определить все возможные входные данные, соответствующие им ожидаемые результаты. При этом будут тестироваться следующие функции:

- *get_network_address* измодуля*collect*
- *get_wlan_name* измодуля*collect*
- *connect* из модуля *collect*
- *collect* из модуля *collect*
- *parse_data* из модуля *parse*

- *excel* из модуля *mine*
- *get_addresses* из модуля *mine*
- *count_vendor_devices* измодуля *mine*

Тесты будут проводиться с помощью библиотек PyUnit [5], а также тестировщиком вручную, когда автоматизировать тестирование невозможно или нецелесообразно.

Автоматические тесты с помощью PyUnit включают в себя определение действий, производимых до теста, определение входных данных, тестирование функций с помощью утверждений (проверка выходных значений или выброс исключений), действия после теста.

Тесты, выполняемые вручную, включают действия или проверки, которые невозможно выполнить автоматически. При этом, каждый для каждого такого теста определяется алгоритм действий и набор проверок.

2.4. Стратегия интеграционного тестирования

Для проведения интеграционного тестирования необходимо определить все возможные входные данные, соответствующие им ожидаемые результаты. Тестирование будет производиться с применением стратегии восходящего тестирования и с помощью библиотеки PyUnit.

Интеграция со сторонними библиотеками manuf, openpyxl и jsonschema отдельно тестироваться не будет, т. к. покрывается блочными тестами.

Интеграционные тесты разделены на группы, каждая из которых тестирует свой порядок интеграции модулей системы:

1. группа тестов взаимодействия модулей collect и parse, а именно функции collect.collect и функции parse.parse_data. При данном взаимодействии функция parse.parse_data вызывается после завершения работы функции collect
2. группа тестов взаимодействия модулей parse и mine, а именно функции parse_data и функций модуля mine, перечисленных в пункте 2.3. При данном взаимодействии функции модуля mine вызываются после

отработки функции parse_data, в них передается возвращаемое значение parse_data

3. группа тестов взаимодействия модуля wifi_stat с одной стороны и группы модулей collect, parse, mine с другой стороны. При этом тестируется разбор аргументов командной строки (единственная функция wifi_stat) и их применение при вызове функций других модулей. При данном взаимодействии функции модулей collect, parse, mine вызываются из модуля wifi_stat после завершения разбора аргументов командной строки

2.5. Стратегия аттестационного тестирования

Аттестационное тестирование проводится для всей системы, что подразумевает выполнение действий в пользовательском интерфейсе командной строки. Для проведения тестирования в данном случае необходим терминал и интерпретатор Python 3. При проведении аттестационного тестирования будет проверяться набор функциональностей системы, доступный пользователю:

1. периодический опрос локальной беспроводной сети (в которой находится компьютер с запущенным приложением),формирование и сохранение на диск JSON-файлов с отчетами о сканировании;
2. возможность одновременного сканирования нескольких сетей;
3. формирование по собранным в рамках опроса сети данным файлов MSExcel, содержащих таблицы с количеством активных пользователей и распределению устройств по производителям оборудования в каждой из сетей;

2.6. Критерии прохождения тестов

Тест считается пройдённым, если конечный результат соответствует ожидаемому результату. Ожидаемый результат - это результат соответствующий ожидаемой работе функции (в соответствии с ее спецификацией). Ожидаемый результат должен присутствовать в описании каждого теста в плане тестирования.

2.7. Критерии приостановления тестирования

Тестирование должно быть приостановлено, если количество непройденных тестов превысит 10% от их общего количества. После этого тестировщик должен проанализировать результат и сообщить об ошибке, либо признать новый результат корректным (ожидаемым). После этого может быть принято решение об исправлении либо программы, либо результата.

2.8. Критерии возобновления работы

Необходимо заново начать тестирование при получении уведомления, что найденные при тестировании ошибки исправлены. Повторное тестирование необходимо начинать с самого начала.

3. Детальный план тестов

3.1. Блочные тесты

Способ запуска блочного теста определяется в графе «Метод», способы запуска и требования к ним перечислены в п. 2.3.

3.1.1. Модуль c*ollect*

1. Объект: *get_network_address*

Тип: общий

Описание: проверка корректности определения адреса подсети в условиях подключения к сети IT-GUEST.

Косвенные входные данные: вывод стандартной утилиты командной строки Windows – Get-NetIPAddress следующего типа:

IPAddress : 172.20.44.100

InterfaceIndex : 2

InterfaceAlias : Беспроводная сеть

AddressFamily : IPv4

Type : Unicast

PrefixLength : 22

PrefixOrigin : Dhcp

SuffixOrigin : Dhcp

AddressState : Preferred

ValidLifetime : Infinite ([TimeSpan]::MaxValue)

PreferredLifetime : Infinite ([TimeSpan]::MaxValue)

SkipAsSource : False

PolicyStore : ActiveStore

Выходные данные: Адрес+маска текущей подсети компьютера в виде Python объекта ipaddress.IPv4Network

Ожидаемый результат: возвращение адреса подсети IT-GUEST: 172.20.44.0/22

Метод: выполняется вручную

2. Объект: *get_network_address*

Тип: негативный

Описание: проверка корректности реакции функции на отсутствие подключения к сети.

Косвенные входные данные: вывод стандартной утилиты командной строки Windows – Get-NetIPAddress

Ожидаемый результат: выброс исключения

Метод: черного ящика, выполняется вручную

3. Объект: get_network_address

Тип: краевой

Описание: проверка корректности определения адреса подсети с маской /31 (255.255.255.252) и адресом 192.168.0.1 (настраиваются перед тестированием)

Косвенные входные данные: вывод стандартной утилиты командной строки Windows – Get-NetIPAddress следующего типа:

IPAddress : 192.168.0.2

InterfaceIndex : 2

InterfaceAlias : Беспроводная сеть

AddressFamily : IPv4

Type : Unicast

PrefixLength : 30

PrefixOrigin : Dhcp

SuffixOrigin : Dhcp

AddressState : Preferred

ValidLifetime : Infinite ([TimeSpan]::MaxValue)

PreferredLifetime : Infinite ([TimeSpan]::MaxValue)

SkipAsSource : False

PolicyStore : ActiveStore

Выходные данные: Адрес+маска текущей подсети компьютера в виде Python объекта ipaddress.IPv4Network

Ожидаемый результат: получение правильного адреса подсети (192.168.0.1/31)

Метод: черного ящика, выполняется вручную

4. Объект: get_network_address

Тип: краевой

Описание: проверка корректности определения адреса подсети с маской /1 (128.0.0.0) и адресом 192.168.0.1 (настраиваются перед тестированием)

Косвенные входные данные: вывод стандартной утилиты командной строки Windows – Get-NetIPAddress следующего типа:

IPAddress : 192.168.0.2

InterfaceIndex : 2

InterfaceAlias : Беспроводная сеть

AddressFamily : IPv4

Type : Unicast

PrefixLength : 1

PrefixOrigin : Dhcp

SuffixOrigin : Dhcp

AddressState : Preferred

ValidLifetime : Infinite ([TimeSpan]::MaxValue)

PreferredLifetime : Infinite ([TimeSpan]::MaxValue)

SkipAsSource : False

PolicyStore : ActiveStore

Выходные данные: Адрес+маска текущей подсети компьютера в виде Python объекта ipaddress.IPv4Network

Ожидаемый результат: получение правильного адреса подсети (192.168.0.1/1)

Метод: черного ящика, выполняется вручную

5. Объект: *get_wlan_name*

Тип: общий

Описание: проверка корректности определения имени беспроводной сети в условиях подключения к сети IT-GUEST.
Косвенные входные данные: вывод стандартной утилиты командной строки Windows – netsh
Выходные данные: имя сети в виде строки
Ожидаемый результат: будет возвращено правильное имя сети IT_GUEST
Метод: черного ящика, выполняется вручную

6. Объект: *get_wlan_name*

Тип: негативный
Описание: проверка корректности реакции функции на отсутствие подключения к сети.
Косвенные входные данные: вывод стандартной утилиты командной строки Windows – netsh
Выходные данные: имя сети в виде строки
Ожидаемый результат: будет возвращен нулевой указатель (None)
Метод: черного ящика, выполняется вручную

7. Объект: *get_wlan_name*

Тип: специальный
Описание: проверка корректности определения имени беспроводной сети на кириллице в условиях подключения к сети. Имя сети – сеть123 – настраивается перед тестированием.
Косвенные входные данные: вывод стандартной утилиты командной строки Windows – netsh
Выходные данные: имя сети в виде строки
Ожидаемый результат: будет возвращено правильное имя сети – сеть123
Метод: черного ящика, выполняется вручную

8. Объект: *connect*

Тип: общий
Описание: проверка успешности подключения компьютера к беспроводной сети при отсутствии подключения до вызова функции connect

Входные данные: имя беспроводной сети для подключения - IT-GUEST

Ожидаемый результат: компьютер будет подключен к сети с указанным именем

Метод: выполняется вручную

9. Объект: *connect*

Тип: общий

Описание: проверка успешности подключения компьютера к беспроводной сети IT-GUEST при наличии подключения к другой сети (PSU-STUD) до вызова функции connect

Входные данные: имя беспроводной сети для подключения IT-GUEST

Ожидаемый результат: компьютер будет отключен от текущей сети и подключен к сети с указанным именем

Метод: черного ящика, выполняется вручную

10. Объект: *connect*

Тип: негативный

Описание: проверка работы функции подключения к сети при отсутствии сети с передаваемым функции именем

Входные данные: имя несуществующей беспроводной сети (daskjg124) для подключения

Ожидаемый результат: компьютер не будет подключен к сети

Метод: черного ящика, выполняется вручную.

11. Объект: *parse_nmap_xml*

Тип: общий

Описание: проверка корректности разбора XML-вывода утилиты Nmap

Входные данные: строка, содержащая XML-структуру, генерируемую утилитой Nmap при сканировании сети следующего типа:

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE nmaprun>

<?xml-stylesheet href="file:///C:/tools/Nmap/nmap.xsl" type="text/xsl"?>

<!-- Nmap 7.31 scan initiated Sun Dec 11 11:55:19 2016 as:

C:\\tools\\Nmap\\nmap.exe -PR -sP -oX - 192.168.0.0/24 -->

<nmaprun scanner="nmap" args="C:\\tools\\Nmap\\nmap.exe -PR -sP -oX -
192.168.0.0/24" start="1481446519" startstr="Sun D
ec 11 11:55:19 2016" version="7.31" xmloutputversion="1.04">
<verbose level="0"/>
<debugging level="0"/>
<host><status state="up" reason="arp-response" reason_ttl="0"/>
<address addr="192.168.0.1" addrtype="ipv4"/>
<address addr="C0:4A:00:F0:3E:AE" addrtype="mac" vendor="Tp-link
Technologies"/>
<hostnames>
</hostnames>
<times srtt="12125" rttvar="11500" to="100000"/>
</host>
<host><status state="up" reason="arp-response" reason_ttl="0"/>
<address addr="192.168.0.101" addrtype="ipv4"/>
<address addr="40:F0:2F:18:DF:18" addrtype="mac" vendor="Liteon Technology"/>
<hostnames>
</hostnames>
<times srtt="5000" rttvar="5000" to="100000"/>
</host>
<host><status state="up" reason="localhost-response" reason_ttl="0"/>
<address addr="192.168.0.100" addrtype="ipv4"/>
<hostnames>
</hostnames>
</host>
<runstats><finished time="1481446524" timestr="Sun Dec 11 11:55:24 2016"
elapsed="4.15" summary="Nmap done at Sun Dec 11
11:55:24 2016; 256 IP addresses (3 hosts up) scanned in 4.15 seconds"
exit="success"/><hosts up="3" down="253" total="2
56"/>

</runstats>

</nmaprun>

Выходные данные: ассоциативный массив, ключами которого являются IP-адреса, а значениями – соответствующие им MAC-адреса.

Ожидаемый результат: будет возвращен ассоциативный массив, содержащий все записи IP-MAC, имеющиеся в исходной XML-структуре. В данном случае:

192.168.0.1 -> C0:4A:00:F0:3E:AE

192.168.0.101 → 40:F0:2F:18:DF:18

Метод: черного ящика, выполняется с помощью PyUnit

12. Объект: *parse_nmap_xml*

Тип: общий

Описание: проверка корректности разбора XML-вывода утилиты Nmap

Входные данные: строка, содержащая XML-структуру, генерируемую утилитой Nmap при сканировании сети, в которой для одного из компьютеров указан IP-адрес, но не указан MAC-адрес (см. пример выше)

Выходные данные: ассоциативный массив, ключами которого являются IP-адреса, а значениями – соответствующие им MAC-адреса.

Ожидаемый результат: будет возвращен ассоциативный массив, содержащий все записи IP-MAC, имеющиеся в исходной XML-структуре, кроме той, в которой отсутствует MAC-адрес (см. пример выше)

Метод: черного ящика, выполняется с помощью PyUnit

13. Объект: *parse_nmap_xml*

Тип: негативный

Описание: проверка реакции функции разбора XML-вывода утилиты Nmap на некорректные входные данные

Входные данные: строка, содержащая XML-структуру, не являющуюся структурой, генерируемую утилитой Nmap при сканировании сети, например

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE unknown>

<sometag><someothertag>shgsh</someothertag></sometag>

Выходные данные: ассоциативный массив, ключами которого являются IP-адреса, а значениями – соответствующие им MAC-адреса.

Ожидаемый результат: будет возвращен пустой ассоциативный массив

Метод: черного ящика, выполняется с помощью PyUnit

14. Объект: *parse_nmap_xml*

Тип: негативный

Описание: проверка реакции функции разбора XML-вывода утилиты Nmap на некорректные входные данные

Входные данные: строка, содержащая невалидную XML-структуру, например

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE unknown>

<sometag><someothertag>shgsh

Выходные данные: ассоциативный массив, ключами которого являются IP-адреса, а значениями – соответствующие им MAC-адреса.

Ожидаемый результат: будет возвращен пустой ассоциативный массив

Метод: черного ящика, выполняется с помощью PyUnit

3.1.2. Модуль *parse*

Для проверки корректности данных в читаемом JSON-файле в модуле *parse* используется валидация JSON-файлов с помощью JSON Schema используя библиотеку Python jsonschema. Схемаследующеговида:

```
{"type": "object",
"properties": {
"ssid": {"type": "string"},
"started": {"type": "string",
"pattern": "^[\d]{2}\.[\d]{2}\.[\d]{2} [\d]{2}\:[\d]{2}\:[\d]{2}$"},
"finished": {"type": "string",
"pattern": "^[\d]{2}\.[\d]{2}\.[\d]{2} [\d]{2}\:[\d]{2}\:[\d]{2}$"},
"address": {"type": "string",
"pattern": "^(?:[\d]{1,3}\.){3}[\d]{1,3}/[\d]{2}$"},
"hosts": {"type": "object",
```

"patternProperties": {
"^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$":
{"type": "string",
"pattern": "^([0-9A-Fa-f]{2}[:-]){5}([0-9A-Fa-f]{2})$"}
},
"additionalProperties": False},
"active": {"type": "number", "minimum": 0}},
"required": ["ssid", "started", "finished", "hosts", "active"]
}

Файлы содержат JSON-объекты со следующими полями: ssid – имя беспроводной сети, started – дата начала опроса, finished – дата окончания опроса, address – адрес подсети, active – число активных устройств, hosts – объект, ключами которого являются IP-адреса активных устройств, а значениями – их MAC-адреса.

Функция parse_data формирует ассоциативный массив следующего вида:

- Имя беспроводной сети
  - дата окончания опроса сети
    - ассоциативный массив, соответствующий содержимому JSON-файла

1. Объект: *parse_data*

Тип: общий

Описание: проверка успешности чтения JSON-файлов с диска и формирования массива с данными

Входные данные: список сетей (IT-GUEST, PSU-STUD), для которых нужно прочитать данные, имя папки на диске (./data), в которой находятся файлы

Косвенные входные данные: содержимое JSON-файлов указанного выше формата

Выходные данные: ассоциативный массив указанного выше вида

Ожидаемый результат: выходной ассоциативный массив будет содержать все записи, соответствующие прочитанным файлам

Выполняется с помощью PyUnit

2. Объект: *parse_data*

Тип: негативный

Описание: проверка реакции на отсутствие поддиректории с именем, соответствующим одной их переданных функции сетей, в директории с данными

Входные данные: список сетей, для которых нужно прочитать данные (для одного из имен данные на диске отсутствуют) (PSU-STUD, adgad123, Для последнего данные отсутствуют), имя папки на диске (./data), в которой находятся файлы

Косвенные входные данные: содержимое JSON-файлов указанного выше формата

Выходные данные: ассоциативный массив указанного выше вида

Ожидаемый результат: выходной ассоциативный массив будет содержать все записи, соответствующие прочитанным файлам, а значением для ключа верхнего уровня, содержащем имя отсутствующей сети, будет пустой ассоциативный массив.

Выполняется с помощью PyUnit

3. Объект: *parse_data*

Тип: общий

Описание: проверка реакции функции на отсутствие директории с данными

Входные данные: список сетей (PSU-STUD, IT-GUEST), для которых нужно прочитать данные, имя отсутствующей папки на диске (./adgsdag), в которой должны находятся файлы

Выходные данные: ассоциативный массив указанного выше вида

Ожидаемый результат: выходной ассоциативный массив будет пустым

Метод: черного ящика, выполняется с помощью PyUnit

4. Объект: *parse_data*

Тип: негативный

Описание: проверка реакции на некорректное содержимое файла

Входные данные: список сетей (PSU-STUD, IT-GUEST), для которых нужно прочитать данные, имя папки на диске (./data), в которой находятся файлы
Косвенные входные данные: содержимое JSON-файлов, один из которых не соответствует указанной выше схеме
Выходные данные: ассоциативный массив указанного выше вида
Ожидаемый результат: выходной ассоциативный массив будет содержать все записи, соответствующие прочитанным файлам, кроме того файла, который содержит некорректную структуру
Метод: черного ящика, выполняется с помощью PyUnit

3.1.3. Модуль *mine*

1. Объект: *excel*

Тип: общий
Описание: проверка корректности создания файла MSExcel по набору данных
Входные данные: ассоциативный массив с данными в указанном выше (в *parse*) формате, содержащий 2 сети (IT-GUEST, PSU-STUD), 10 записей на каждую из сетей
Выходные данные: файл “wifi.xlsx”
Ожидаемый результат: на диске будет создан файл “wifi.xlsx” с 4 листами, по два на каждую из сетей (с данными об активности и данными о производителях оборудования)
Метод: черного ящика, выполняется вручную

2. Объект: *excel*

Тип: негативный
Описание: проверка поведения функции при передаче в нее некорректных данных
Входные данные: ассоциативный массив, не соответствующий указанному выше формату
Ожидаемый результат: возникновение исключения ValueError
Метод: черного ящика, выполняется с помощью PyUnit

3. Объект: get_addresses

Тип: общий

Описание: проверка корректности определения множества уникальных MAC-адресов

Входные данные: ассоциативный массив в указанном выше формате, имя сети (IT-GUEST), в которой нужно выделить адреса

Выходные данные: множество с MAC-адресами

Ожидаемый результат: будет получено множество, содержащее все имеющиеся MAC адреса в переданных данных

Метод: черного ящика, выполняется с помощью PyUnit

4. Объект: get_addresses

Тип: негативный

Описание: проверка реакции функции на некорректные данные

Входные данные: ассоциативный массив не в указанном формате, имя сети (IT-GUEST), в которой нужно выделить адреса

Ожидаемый результат: возникновение исключения ValueError

Метод: черного ящика, выполняется с помощью PyUnit

5. Объект: get_addresses

Тип: негативный

Описание: проверка реакции функции на некорректные данные

Входные данные: ассоциативный массив в указанном выше формате, имя сети (IT-GUEST), в которой нужно выделить адреса, отсутствующей во входных данных

Выходные данные: множество с MAC-адресами

Ожидаемый результат: будет получено пустое множество

Метод: черного ящика, выполняется с помощью PyUnit

6. Объект: count_vendor_devices

Тип: общий

Описание: проверка корректности определения распределения устройств по производителям

Входные данные: ассоциативный массив в указанном выше формате, имя сети (IT-GUEST), в которой нужно выделить производителей, имя файла Wireshark с базой производителей (./manuf.txt)
Выходные данные: Ассоциативный массив, в котором ключи — названия производителей, значения – количество встреч этих производителей в сети
Ожидаемый результат: будет получен ассоциативный массив с распределением производителей
Метод: черного ящика, выполняется с помощью PyUnit

7. Объект: count_vendor_devices

Тип: негативный
Описание: проверка реакции функции на некорректные данные
Входные данные: ассоциативный массив в указанном выше формате, имя сети (adgadg123), в которой нужно выделить производителей, отсутствующей во входных данных, имя файла Wireshark с базой производителей (./manuf.txt)
Выходные данные: множество с MAC-адресами
Ожидаемый результат: будет получен пустой ассоциативный массив
Метод: черного ящика, выполняется с помощью PyUnit

8. Объект: count_vendor_devices

Тип: негативный
Описание: проверка реакции функции на некорректные данные
Входные данные: ассоциативный массив в указанном выше формате, имя сети (IT-GUEST), в которой нужно выделить производителей, имя отсутствующего на диске файла Wiresharkc базой производителей (./sdljg)
Ожидаемый результат: возникновение исключения ValueError
Метод: черного ящика, выполняется с помощью PyUnit

9. Объект: count_vendor_devices

Тип: негативный
Описание: проверка реакции функции на некорректные данные

Входные данные: ассоциативный массив не в указанном формате, имя сети (IT-GUEST), в которой нужно выделить производителей, имя файла Wireshark с базой производителей (./manuf.txt)

Ожидаемый результат: возникновение исключения ValueError

Метод: черного ящика, выполняется с помощью PyUnit

3.2. Интеграционные тесты

Шаги интеграции см. в п. 2.4.

3.2.1. Модули collect и parse

1. Объекты: функции collect.collect и parse.parse_data

Тип: общий

Описание: проверка корректности обработки файлов, создаваемых функцией collect в функции parse_data

Алгоритм: 1) вызвать функцию collect в окружении сети PSU-STUD ГК ПетрГУ

2) дождаться окончания сборки данных в сети записи на диск JSON-файла с данными о сканировании

3) вызвать функцию parse_data для файла, созданного на предыдущем шаге

4) сравнить возвращаемое значение функции parse_data с содержимым файла, созданного на шаге 2

Ожидаемый результат: возвращаемое значение функции parse_data будет содержать ассоциативный массив указанной в пункте 3.1.2 структуры, в котором будут все данные, находящиеся в файле из шага 2

2. Объекты: функции collect.collect и parse.parse_data

Тип: негативный

Описание: проверка корректности обработки файлов, создаваемых функцией collect в функции parse_data при аварийном завершении работы функции collect

Алгоритм: 1) вызвать функцию collect в окружении сети PSU-STUD ГК ПетрГУ

2) прервать выполнение функции collect

3) вызвать функцию parse_data для целевой директории collect для файлов

4) убедиться в отсутствии исключений функции parse_data

5) проверить возвращаемое значение функции parse_data на соответствие определенному формату

Ожидаемый результат: исключения функции parse_data будут отсутствовать, возвращаемое значение функции parse_data будет соответствовать формату, указанному в пункте 3.1.2

3.2.2. Модули parse и mine

При тестировании взаимодействия модулей parse и mine на диске имеются набор файлов JSON указанного в п. 3.1.2 формата с данными, собранными в сети. Функции parse_data во всех тестах указывается местоположение этих файлов.

1. Объекты: функции parse.parse_data и mine.excel

Тип: общий

Описание: проверка корректности переноса данных в формат excel

Алгоритм: 1) вызвать функцию parse_data

2) возвращаемое значение функции parse_data передать функции excel

3) сравнить возвращаемое значение функции parse_data и содержимого файла wifi.xlsx, созданного функцией excel

Ожидаемый результат: файл будет содержать все данные, имеющиеся в возвращаемом значении parse_data, в формате MSExcel

2. Объекты: функции parse.parse_data и mine.excel

Тип: негативный

Описание: проверка корректности переноса данных в формат excel при отсутствии данных на диске

Алгоритм: 1) вызвать функцию parse_data в условиях отсутствия на диске файлов JSON

2) возвращаемое значение функции parse_data передать функции excel

3) сравнить возвращаемое значение функции parse_data и содержимого файла wifi.xlsx, созданного функцией excel

Ожидаемый результат: будет создан пустой файл формата MSExcel

3. Объекты: функции parse.parse_data и mine.get_addresses

Тип: общий

Описание: проверка корректности выделения множества MAC-адресов

Алгоритм: 1) вызвать функцию parse_data

2) возвращаемое значение функции parse_data передать функции get_addresses

3) сравнить возвращаемые значения функцийparse_data и get_addresses

Ожидаемый результат: возвращаемое значение get_addresses будет содержать все адреса, имеющиеся в возвращаемом значении parse_data

4. Объекты: функции parse.parse_data и mine.get_addresses

Тип: негативный

Описание: проверка корректности выделения множества MAC-адресов в условиях отсутствия данных

Алгоритм: 1) вызвать функцию parse_data в условиях отсутствия на диске JSON-файлов

2) возвращаемое значение функции parse_data передать функции get_addresses

3) сравнить возвращаемые значения функций parse_data и get_addresses

Ожидаемый результат: возвращаемое значение get_addresses будет является пустым множеством

5. Объекты: функции parse.parse_data и mine.count_vendor_devices

Тип: общий

Описание: проверка корректности выделения статистики популярности производителей устройств

Алгоритм: 1) вызвать функцию parse_data

2) возвращаемое значение функции parse_data передать функции count_vendor_devices

3) сравнить возвращаемые значения функций parse_data и count_vendor_devices

Ожидаемый результат: возвращаемое значение count_vendor_devices будет содержать статистику соответствующую возвращаемому значению parse_data

6. Объекты: функции parse.parse_data и mine.count_vendor_devices

Тип: негативный

Описание: проверка корректности выделения статистики популярности производителей устройств в условиях отсутствия данных

Алгоритм: 1) вызвать функцию parse_data в условиях отсутствия на диске файлов JSON

2) возвращаемое значение функции parse_data передать функции count_vendor_devices

3) сравнить возвращаемые значения функций parse_data и count_vendor_devices

Ожидаемый результат: возвращаемое значение count_vendor_devices будет содержать пустой ассоциативный массив

3.2.3. Модуль wifi_stat и группа модулей collect, parse, mine

Модули collect, parse, mine в этой группе подключались к wifi_stat вместе.

Во все тестах данного типа проверяется правильность вызова функций после распознавания аргументов командной строки. В функциях модулей collect, parse и mine добавляется вывод значений аргументов функций, проводится сравнение их с введенными аргументами командной строки. В модуле wifi_stat аргументы распознаются с помощью стандартной библиотеки argparse языка Python. Список используемых аргументов следующий:

- -o [вызываемой функции] имя функции, которая будет вызвана. Возможные значения: collect, excel
- -n [сеть1 сеть2 ...]: список сетей для опроса в функции collect
- -d [имя_каталога]: имя каталога, куда сохраняются и откуда читаются JSON-файлы
- -x [имя_файла]: имя файла MSExcel, который будет использоваться в функции mine.excel

1. Объекты: модуль wifi_stat и функция collect.collect

Тип: общий

Описание: проверка корректности использования аргументов при вызове функции collect

Алгоритм: 1) в командной строке модуль wifi_stat вызывается с помощью команды «pythonwifi_stat.py -ocollect -nPSU-STUDPSU-STAFF -ddata1»
2) в функции collect проверяется соответствие переданных функции параметров аргументам командной строки
Ожидаемый результат: параметр collect под именем *nets*будет содержать список строк [“PSU-STUD”, “PSU-STAFF”], а аргумент под именем *data_dir* будет содержать строку «data1»

2. Объекты: модуль wifi_stat и функция collect.collect

Тип: негативный
Описание: проверка корректности вызова функции collect при отсутствии аргументов командной строки
Алгоритм: 1) в командной строке модуль wifi_stat вызывается с помощью команды «pythonwifi_stat.py -ocollect»
2) в функции collect проверяется соответствие переданных функции параметров аргументам командной строки
Ожидаемый результат: параметр collect под именем nets будет содержать значение по умолчанию – список строк [“PSU-STUD”, “PSU-STAFF”, “IT-GUEST”], аргумент под именем *data_dir* будет содержать значение по умолчанию – строку «data»

3. Объекты: модуль wifi_stat и функции parse.parse_data, mine.excel

Тип: общий
Описание: проверка корректности использования аргументов при вызове функции excel
Алгоритм: 1) в командной строке модуль wifi_stat вызывается с помощью команды «pythonwifi_stat.py -oexcel -ddata1 -xtest.xslx»
2) в функции parse_data проверяется соответствие переданных функции параметров аргументам командной строки
3) в функции excel проверяется соответствие переданных функции параметров аргументам командной строки

Ожидаемый результат: параметр parse_data под именем *data_dir* будет содержать строку «data1», параметр excel под именем *excel_file*будет содержать строку «test.xlsx»

4. Объекты: модуль wifi_stat и функции parse.parse_data, mine.excel

Тип: негативный

Описание: проверка корректности вызова функций parse_data и excel при отсутствии аргументов командной строки

Алгоритм: 1) в командной строке модуль wifi_stat вызывается с помощью команды «pythonwifi_stat.py -oexcel»

2) в функции parse_data проверяется соответствие переданных функции параметров аргументам командной строки

3) в функции excel проверяется соответствие переданных функции параметров аргументам командной строки

Ожидаемый результат: параметр parse_data под именем *data_dir* будет содержать значение по умолчанию – строку «data», параметр excel под именем *excel_file* будет содержать значение по умолчанию – строку «wifi.xlsx»

3.3. Аттестационные тесты

Набор функций приложения, которые проверяются в аттестационном тестировании, определен в п. 2.5.

При работе в реальных сетях количество файлов JSON, появляющихся на диске по мере опроса за определенный промежуток времени, изменяется в зависимости от текущей нагрузки на сеть: при большей нагрузке тратится больше времени на опрос сети. При максимальной нагрузке на любую сеть университета (1024 активных устройства) за 30 минут опроса на диске создается не менее 5 файлов.

Тесты для проверки функции 1 (Периодическое сканирование сети и формирование отчета о сканировании):

1. Периодическое сканирование активности в сети

Тип теста: общий

Алгоритм: 1) запустить приложение для опроса одной сети (PSU-STUD)

2) подождать 30 минут

Ожидаемый результат: на диске будет создано от 5 до 10 файлов JSON (в зависимости от текущей нагруженности сети) с данными об активности. Критерий количества файлов см. выше.

2. Сканирование несуществующей сети

Тип теста: негативный

Алгоритм: 1) запустить приложение для опроса несуществующей сети (lsdgsd1124)

2) подождать 30 минут

Ожидаемый результат: на диске не будет создано файлов JSON.

Тесты для проверки функции 2 (возможность сканирования нескольких сетей):

3. Периодическое сканирование активности в нескольких сетях

Тип теста: общий

Алгоритм: 1) запустить приложение для опроса двух сетей (PSU-STUD, IT-GUEST)

2) подождать 30 минут

Ожидаемый результат: на диске будет создано от 5 до 10 файлов JSON для каждой из сканируемых сетей. Критерий количества файлов см. выше.

4. Получение информации о сетях, исследуемых в рамках магистерской диссертации

Тип теста: специальный

Алгоритм: 1) запустить приложение для опроса сетей PSU-STUD, PSU-STAFF, IT-GUEST;

2) подождать 1 час;

3) остановить приложение;

4) убедиться, что на диске появились JSON-файлы с данным для всех указанных сетей

5) запустить приложение для формирования файла MSExcel по собранным данным

Ожидаемый результат: на диске будет создан файл MSExcel с данным о сканировании указанных сетей в указанный период.

Тесты для проверки функции 3 (формирование отчета о сканировании в файле MS Excel):

5. Формирование файла MSExcel по собранным данным

Тип теста: общий

Алгоритм: 1) запустить приложение для опроса сети;

2) подождать 30 минут;

3) остановить приложение;

4) убедиться, что на диске появились JSON-файлы с данным

5) запустить приложение для формирования файла MSExcel по собранным данным

Ожидаемый результат: на диске будет создан файл MSExcel с данным о сканировании сети в указанный период.

6. Попытка формирования файла MSExcel без собранных данных

Тип теста: негативный

Алгоритм: 1) убедиться в отсутствии на диске в каталоге с данными приложенияJSON-файлов с данным

2) запустить приложение для формирования файла MSExcel по собранным данным

Ожидаемый результат: на диске не будет создан файл MSExcel, будет выведено сообщение об отсутствии требуемых файлов.

3.4. Дополнительные виды тестирования

Тест для проверки функции 1

1. Сканирование сети под высокой нагрузкой

Тип теста: нагрузочный

Алгоритм:

1) Запустить приложение для опроса сети PSU-STUD в понедельник, 15:00 (время максимальной нагрузки по статистике от системых администраторов ПетрГУ)

2) Дождаться окончания опроса сети

Ожидаемый результат: опрос сети будет завершен не более чем за 5 минут, на диске появится JSON-файл с результатами опроса

3.5. Пример реализации теста

Далее приведен пример реализации автоматического блочного теста с помощью PyUnit.Реализуется тест 3.1.3.3 в модуле *mine* для функции get_addresses. Код теста (на языке Python) приведен на рис. 2. Первые две строчки — это подключение библиотеки тестирования и тестируемой функции. Сам тест обернут (согласно документации библиотеки) в класс-наследник класса TestCase. В методе *setU p*создаваемого класса происходит инициализация свойства *input_data*, содержащего входные данные для теста в указанном в разделе 3.1.2 формате (в виде ассоциативного массива). В свойство *await_data* записывается ожидаемое значение (в виде множества). В методе *testCommon* происходит вызов функции get_addresses с передачей в нее входных данных и имени сети; сравнение выходных данных с ожидаемым значением; проверка успешности сравнения с помощью утверждения (функция assertEquals). Третьим аргументом функции assertEqual является сообщение об ошибки пре неудаче сравнения.

Остальные автоматические тесты реализованы аналогично.

Рисунок 2 Пример реализации теста

```
import unittest
from mine import get_addresses

class GetAddressesTestCase(unittest.TestCase):
    def setUp(self):
        self.input_data = \
        {"test_net":
            {"10.10.2016": {
                "hosts": {"172.20.44.172": "11:11:11:11:11:11",
                          "172.20.46.55":  "22:22:22:22:22:22"},
                "started": "16.10.10 14:05:50",
                "ssid": "test_net",
                "active": 2,
                "address": "172.20.44.0/22",
                "finished": "16.10.10 14:06:23"}
            },
            {"11.10.2016": {
                "hosts": {"172.20.44.172": "11:11:11:11:11:11",
                          "172.20.47.227": "33:33:33:33:33:33"},
                "started": "16.10.11 14:05:50",
                "ssid": "test_net",
                "active": 2,
                "address": "172.20.44.0/22",
                "finished": "16.10.10 14:06:23"}
            }
        }

        self.await_data = {"11:11:11:11:11:11",
                           "22:22:22:22:22:22"
                           "33:33:33:33:33:33"}

    def test_get_addresses(self):
        self.assertEquals(get_addresses(self.input_data, "test_net"),
                          self.await_data,
                          'Полученное значение не совпадает с ожидаемым')
```

3.6. Оценка покрытия кода

Ввиду сложности современного программного обеспечения становится невозможно добиться 100% покрытия тестами. Кроме того, некоторые тесты невозможно автоматизировать, что сокращает долю кода, покрытого автоматическими тестами (хотя код при это протестирован). Расчет доли покрытого тестами кода проводится по формуле Tcov = Ltc/Lcode * 100%, где Tcov – доля кода, покрытого тестами, Ltc – количество строк кода, покрытого тестами, Lcode – общее количество строк кода.

При тестировании покрытия программы wifi_info кроме общей доли покрытия тестами вычислялась так же доля кода, покрытого тестами, который

содержится в функциях, тестируемых автоматически (см. раздел 3). Данный расчет производиться аналогичным образом.

В языке Python для вычисления тестового покрытия используется пакет Coverage.py. В таблице 1 приведен результат вычисления тестового покрытия.

Таблица 1. Тестовое покрытие

Модуль	**Строк кода**	**Общее покрытие**	**Покрытие выделенных функций**
Collect	168	19%	64%
Parse	61	82%	82%
Mine	197	65%	78%
wifi_stat	64	93%	93%
Общее	490	53%	74%

4. Журнал тестирования таблица

Журнал блочного и интеграционного тестирования приложения wifi_stat приведен в таблице 2. Журнал аттестационного тестирования приведен в таблице 3.

Таблица 2. Журнал блочного и интеграционного тестирования

Номера тестов	**Объект(ы)**	**Кол-во тестов**	**Кол-во ошибок**	**Дата проведения**
3.1.1.1, 3.1.1.2, 3.1.1.3, 3.1.1.4	Collect/get_network_address	4	0	18.11.2016
3.1.1.5, 3.1.1.6, 3.1.1.7	Collect/get_wlan_name	3	1 в тесте 3.1.1.5 (см. отчет об ошибке №1)	18.11.2016
3.1.1.8, 3.1.1.9, 3.1.1.10	Collect/connect	3	0	18.11.2016
3.1.1.11, 3.1.1.12, 3.1.1.13, 3.1.1.14	Collect/parse_nmap_xml	4	0	18.11.2016
3.1.2.1, 3.1.2.2, 3.1.2.3, 3.1.2.4	Parse/parse_data	4	0	20.11.2016
3.1.3.1, 3.1.3.2	Mine/excel	2	0	20.11.2016

3.1.3.3, 3.1.3.4, 3.1.3.5	Mine/get_addresses	3	0	20.11.2016
3.1.3.6, 3.1.3.7, 3.1.3.8, 3.1.3.9	Mine/count_vendor_devices	4	1 в тесте 3.1.3.7 (см. отчет об ошибке № 2)	20.11.2016
3.2.1.1, 3.2.1.2	Модули collect и parse	2	0	20.11.2016
3.2.2.1, 3.2.2.2, 3.2.2.3, 3.2.2.4, 3.2.2.5, 3.2.2.6	Модули parse и mine	6	0	20.11.2016
3.2.3.1, 3.2.3.2, 3.2.3.3, 3.2.3.4	Модуль wifi_stat и модули collect, parse, mine	4	0	20.11.2016

Таблица 3. Журнал аттестационного тестирования

Номер теста	**Результат**	**Дата проведения**
3.3.1	пройден	25.11.2016
3.3.2	пройден	10.12.2016
3.3.3	пройден	12.12.2016
3.3.4	пройден	25.11.2016
3.3.5	пройден	10.12.2016
3.3.6	пройден	25.11.2016
3.3.7	пройден	16.12.2016

5. Журнал найденных ошибок

5.1. Отчет об ошибке №1

Тест: 3.1.1.5

Объект тестирования: функция get_wlan_name из модуля connect

Условия: ОС Windows 10 x64, Python 3.5.2 x64, подключение к сети PSU-STUD

Приоритет: критический

Алгоритм: 1) подключиться к сети PSU-STUD 2) вызвать функцию get_wlan_name из консоли python или с помощью теста PyUnit

Ожидаемый результат: функция вернет строку «PSU-STUD»

Фактический результат: функция вернула нулевой указатель (None)

Воспроизводимость: всегда

Дата проведения: 18.11.2016

5.2. Отчет об ошибке №2

Тест: 3.1.3.7

Объект тестирования: функция count_vendor_devices из модуля mine

Условия: ОС Windows 10 x64, Python 3.5.2 x64

Приоритет: средний

Алгоритм: вызвать функцию count_vendor_devices со следующими параметрами:

- data = {“PSU-STAFF”: {“11.11.2016”: {“hosts”: {“192.168.1.1”: “11:11:11:11:11:11”}}}}
- wlan_name = ‘PSU-STUD’

Ожидаемый результат: функция вернет пустой ассоциативный массив

Фактический результат: возникновение исключения AttributeError

Воспроизводимость: всегда

Дата проведения: 20.11.2016

6. Результаты

Результаты, полученные в ходе разработки и выполнения тестов, представлены в таблице 1. В ходе тестирования были обнаружены 2 ошибки – в функциях get_wlan_name и count_vendor_devices. При проведении аттестационного тестирования ошибок обнаружено не было. Найденные ошибки были устранены в рамках отладки, последовавшей за тестированием.

Таблица 4. Результаты тестирования

Объекты	**Количество тестов**	**Кол-во ошибок**
Модуль collect	14	1
Модуль parse	4	0
Модуль mine	9	1
Модули collect и parse	2	0
Модули parse и mine	6	0
Модуль wifi_stat и модули	4	0

collect, parse, mine		
Аттестационные тесты	7	0

Таким образом, после проведения тестирования можно сказать, что приложение wifi_stat соответствует предъявленным к нему требованиям.

Список литературы

1. Библиотека openpyxl. URL: https://openpyxl.readthedocs.io/en/default/
2. Библиотека jsonschema. URL: https://github.com/Julian/jsonschema
3. Библиотека manuf. URL: https://github.com/coolbho3k/manuf/
4. Утилита nmap. URL: https://nmap.org/
5. Библиотека PyUnit. URL: https://wiki.python.org/moin/PyUnit