Defensive Python

- 檔案輸入 / 輸出
- 正則表示式 (Regular expression)
- Set 操作
- 網路封包分析

檔案輸入 / 輸出

- Python file operations are all handled by the file object

1

```
filehandle = open("complete file path",mode)
```

OR

2

```
with open("complete file path",mode) as filehandle:
    #code block to process file using file_handle
```

Mode name	Description
r	read-only mode
w	write-only mode
a	append mode
rt	read text and interpret unicode string (\n or \r\n as end of line)
rb	read binary and no interpret any unicode or end of lines

檔案輸入 / 輸出 (常用方法)

```
>>> filehandle=open('example.txt','r')
>>> dir(filehandle)
['__class__', '__delattr__', '__doc__', '__enter__', '__exit__', '__format__', '
__getattribute__', '__hash__', '__init__', '__iter__', '__new__', '__reduce__',
'__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclass
hook__', 'close', 'closed', 'encoding', 'errors', 'fileno', 'flush', 'isatty', '
mode', 'name', 'newlines', 'next', 'read', 'readinto', 'readline', 'readlines',
'seek', 'softspace', 'tell', 'truncate', 'write', 'writelines', 'xreadlines']
>>>
```

- seek(),tell(): 隨機讀檔 / 寫檔
  - seek(offset[, whence]) 設定檔案指針位置 (offset: 偏移量 ,whence: 位置（ 0: 開頭 ,1: 當前 ,2: 結尾 ）)
  - tell() 回傳檔案指針位置
- read(),readline(),readlines(): 讀檔
- write(),writeline(),writelines(): 寫檔
- close(): 關閉檔案

檔案輸入 / 輸出 (範例)

- 讀檔

```
#Read one line at a time
filehandle = open("filepath",'r')
for oneline in filehandle:
        print(oneline)
filehandle.close()

#Read the entire file into a list
filehandle = open("filepath",'r')
listoflines=filehandle.readlines()
filehandle.close()

#Read the entire file into a single string
filehandle = open("filepath",'r')
listoflines=filehandle.read()
filehandle.close()
```

可使用迴圈察看
檔案內容

將檔案中每行文句
轉成 list

將檔案內容
轉成單一字串

檔案輸入 / 輸出 (範例)

- 寫檔

```
#Writing to the file(overwrite the contents)
filehandle = open("filepath",'w')
filehandle.write("write this line.\n")
filehandle.write("write these\nTwo lines\n")
filehandle.close()

#Append to a file
filehandle = open("filepath",'a')
filehandle.write("Add this to file.\n")
filehandle.close()
```

檔案輸入 / 輸出 (範例)

- 讀檔 (Binary)

Python 2

```
Python 2.7.15+ (default, Aug 31 2018, 11:56:52)
[GCC 8.2.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> filehandle = open('/bin/bash').read()
>>> filehandle[:20]
'\x7fELF\x02\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x03\x00>\x00'
>>>
```

Python3

```
Python 3.6.7 (default, Oct 21 2018, 08:08:16)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> filehandle = open('/bin/bash').read()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/lib/python3.6/codecs.py", line 321, in decode
    (result, consumed) = self._buffer_decode(data, self.errors, final)
UnicodeDecodeError: 'utf-8' codec can't decode byte 0x88 in position 40: invalid
 start byte
>>> filehandle = open('/bin/bash',encoding="latin-1").read()
>>> filehandle[:20]
'\x7fELF\x02\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x03\x00>\x00'
>>>
```

檔案輸入 / 輸出（模組功能）

- 讀取 gzip 壓縮檔
- Linux 會自動將 log 檔做壓縮 (gzip)
- python 提供 gzip 模組，可以輕鬆的讀取 gzip 壓縮檔
- 此外也可使用 zlib 讀取 gzip 壓縮檔

```
>>> import gzip
>>> gz = gzip.open("/var/log/syslog.2.gz","rt")
>>> gz.read(40)
'Nov 17 10:21:56 kali rsyslogd:  [origin '
>>> list_of_lines = gz.readlines()
>>> list_of_lines[2]
'Nov 17 10:22:01 kali gnome-software[1472]: failed to call gs_plugin_add_updates
 on fwupd: The name org.freedesktop.fwupd was not provided by any .service files
\n'
>>>
```

```
>>> import zlib
>>> gz =open("/var/log/syslog.2.gz",'rb').read()
>>> gzd =zlib.decompress(gz,zlib.MAX_WBITS|16)
>>> gzd[:40]
'Nov 17 10:21:56 kali rsyslogd:  [origin '
>>> gzd[:50]
'Nov 17 10:21:56 kali rsyslogd:  [origin software="'
>>>
```

檔案輸入 / 輸出（模組功能）

- 可以使用 "os" 模組檔案查找以及察看資料夾內容
  - os.path.exists("filepath") : 判斷檔案是否存在
  - os.listdir("filepath"): 列出資料夾內之內容
  - os.walk("filepath"): 列出資料夾內之內容 (deep)

```
>>> import os
>>> os.path.exists("/root/Documents/day3/example.txt")
True
>>> os.path.exists("/root/Documents/day3/example2.txt")
False
>>> os.listdir("/root/Documents/day3/")
['day3.odp', 'filepath.txt', 'example.txt', '.~lock.day3.odp#']
```

```
>>> import os
>>> for currentdir,listofdirs,listoffiles in os.walk("/root/Documents"):
...   print("[*] current Directory is :",currentdir)
...   print("           has directories:",listofdirs)
...   print("           has files:",listoffiles)
...
('[*] current Directory is :', '/root/Documents')
('\t\thas directories:', ['argpase', 'day3'])
('\t\thas files:', ['.~lock.SANS(backup) (\xe5\x89\xaf\xe6\x9c\xac).odp#', 'SANS
(backup).odp', 'max.py', 'SANS(backup) (\xe5\x89\xaf\xe6\x9c\xac).odp'])
('[*] current Directory is :', '/root/Documents/argpase')
('\t\thas directories:', [])
('\t\thas files:', ['argpase.py'])
('[*] current Directory is :', '/root/Documents/day3')
('\t\thas directories:', [])
('\t\thas files:', ['day3.odp', 'filepath.txt', 'example.txt', '.~lock.day3.odp#
'])
>>>
```

檔案輸入 / 輸出（模組功能）

- 另外也可使用 "glob" 模組檔案查找以及察看資料夾內容
- 與 "os" 模組不同，可使用檔案命名規則尋找檔案

```
>>> import glob
>>> glob.glob("/root/Documents/*.py")
['/root/Documents/max.py']
>>> glob.glob("/root/Documents/*")
['/root/Documents/argpase', '/root/Documents/SANS(backup).odp', '/root/Documents
/max.py', '/root/Documents/day3', '/root/Documents/SANS(backup) (\xe5\x89\xaf\xe
6\x9c\xac).odp']
>>>
```

正則表示式

- 非常重要的技巧
- 模組： Import re

```
>>> import re
>>> input = "my id is E123456789 . Please check it"
>>> str = re.findall("[A-Z][0-9]{9}",input)
>>> str
['E123456789']
>>>
```

正則表示式

- .match(re,data):Start at the beginning of data searching for pattern
- .search(re,data):Match pattern anywhere in data
- .findall(re,data):Find all occurrences of pattern in data

```
>>> import re
>>> x=re.match("th","this is the test")
>>> x
<_sre.SRE_Match object; span=(0, 2), match='th'>
>>> x.group()
'th'
>>> x=re.search("th","this is the test")
>>> x
<_sre.SRE_Match object; span=(0, 2), match='th'>
>>> x.group()
'th'
>>> x=re.findall("th","this is the test")
>>> x
['th', 'th']
```

正則表示式

- .(period):Wildcard for any one character
- \w: Any text character(a-z,A-Z,0-9, and_)-no special characters
- \W: Opposite of \w

```
>>> re.findall("SANS","The SANS Python class rocks")
['SANS']
>>> re.findall(".ython","I python, you Python. we all python")
['python', 'Python', 'python']
>>> re.findall(r"\w\w\w\w\w\w\w\w","*&$H(@$password(*$@BK#@TF")
['password']
>>> re.findall(r"\w\W","Get the last letters.")
['t ', 'e ', 't ', 's.']
>>> re.findall(r"\W","Moves! left$ to{ right.")
['!', ' ', '$', ' ', '{', ' ', '.']
>>> re.findall(r".\W","Moves! left$ to{ right.")
['s!', 't$', 'o{', 't.']
>>> re.findall(r"\W.","Moves! left$ to{ right.")
['! ', '$ ', '{ ']
```

正則表示式

- \d: Match digits(0-9)
- \D: Opposite of \d
- \s: Matches any white-space character(space,tab,newlines)
- \S: Opposite of \s
- [set of characters]: define your own sets of characters
- \b: Border of a word character(Transition \w <-> \W)
- ^: Matches from the start of the search string
- $: Matches to the end of the search string
- \: Escapes special characters; that is,”\.” means it should really find a period

正則表示式

```
>>> import re
>>> re.findall(".","a 1b 2c3")
['a', ' ', '1', 'b', ' ', '2', 'c', '3']
>>> re.findall("\\d","a1b2c3")
['1', '2', '3']
>>> re.findall("\\D","a1b2c3")
['a', 'b', 'c']
>>> re.findall("\\d.","a1b2c3")
['1b', '2c']
>>> re.findall("\\w","a1! b2- c3")
['a', '1', 'b', '2', 'c', '3']
>>> re.findall("\\w\\w","a1b2c3")
['a1', 'b2', 'c3']
>>> re.findall("^\\w\\w","a1b2c3")
['a1']
>>> re.findall("\\w\\w$","a1b2c3")
['c3']
>>> re.findall("\\b\\w\\w\\b","a1b 2c 3")
['2c']
>>> re.findall("\w\s","a 1b 2c3")
['a ', 'b ']
```

正則表示式

- [set of characters]: define your own sets of characters

```
>>> x=re.findall(r"\d\d/\d\d/\d\d","12/25/00 99/99/99")
>>> x
['12/25/00', '99/99/99']
>>> x=re.findall(r"[0-9]/[0-9]/[0-9]","12/25/00 99/99/99")
>>> x
[]
>>> x=re.findall(r"[0-9][0-9]/[0-9][0-9]/[0-9][0-9]","12/25/00 99/99/99")
>>> x
['12/25/00', '99/99/99']
>>> x=re.findall(r"[0-1][0-2]/[0-2][0-5]/[0][0]","12/25/00 99/99/99")
>>> x
['12/25/00']
```

正則表示式

- (?:text1|text2|text3) match text1 or text2 or text3

```
>>> x=re.findall(r"0[1-9]|1[0-2]","12/25/00 99/99/99")
>>> x
['12']
>>> x=re.findall(r"(?:0[1-9]|1[0-2])/(?:0[1-9]|[1-2][0-9]|3[0-1])/\d\d","12/25/0
0 99/99/99 01/19/00")
>>> x
['12/25/00', '01/19/00']
```

正則表示式

- What if you want 100 \d characters? Or a variable number?
- + : One or more of the previous character
- *: Zero or more of the previous
- ?: The previous character is optional(match 0 or 1)
- {x}: Match exactly x copies of the previous character
- {x,y}: Match between x and y of the previous character

正則表示式

```
>>> re.findall(r"\d","123abc")
['1', '2', '3']
>>> re.findall(r"\d+","123abc")
['123']
>>> re.findall(r".\.","Hello. This. Is a test. ok.")
['o.', 's.', 't.', 'k.']
>>> re.findall(r".*\.","Hello. This. Is a test. ok.")
['Hello. This. Is a test. ok.']
```

```
>>> re.findall("(?:0?[1-9]|1[1-2])/(?:0?[1-9]|[1-2][0-9]|3[0-1])/\d\d","13/32/31
 1/8/00")
['1/8/00']
```

正則表示式 (Greedy Matching)

- * and + are greedy! They match as much as they can.
- *?,+?: The? Turns off “greedy” matching

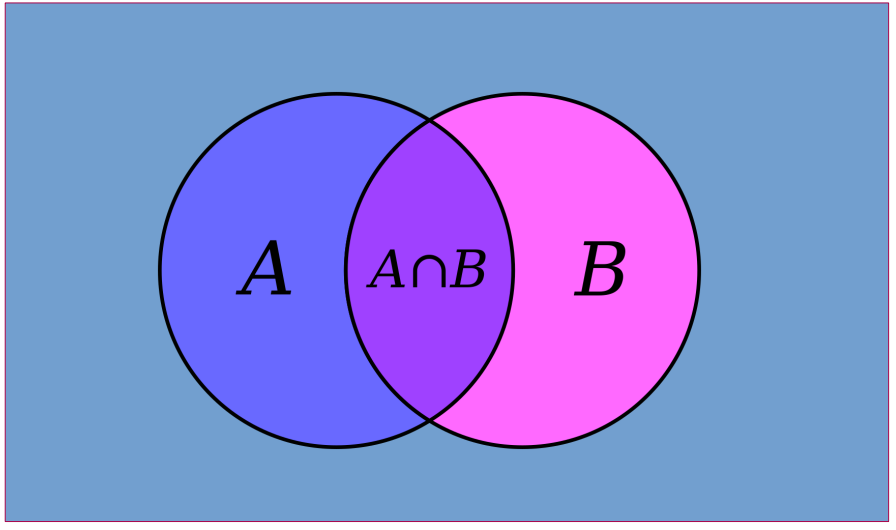
```
>>> x=re.findall(r"[A-Z].+\.","Hello. Hi. Python rocks. I know.")
>>> x
['Hello. Hi. Python rocks. I know.']
>>> x=re.findall(r"[A-Z].+?\.","Hello. Hi. Python rocks. I know.")
>>> x
['Hello.', 'Hi.', 'Python rocks.', 'I know.']
```

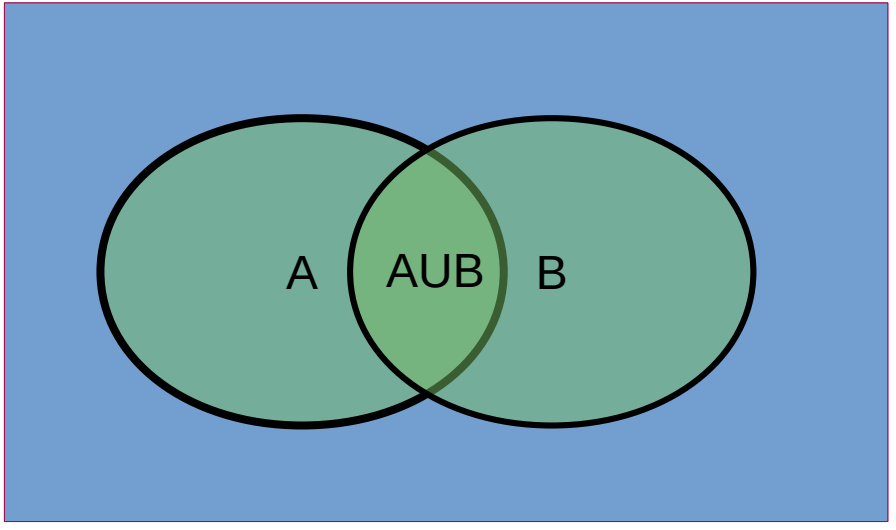
正則表示式 (Not Custom Set)

- [^...] :Match 定義之外的元素

```
>>> x=re.findall(r"[A-Z][^A-Z]+","Things That start with Caps")
>>> x
['Things ', 'That start with ', 'Caps']
>>> x=re.findall(r"[A-Z][^?.!]+","Find. The sentences? Yes!")
>>> x
['Find', 'The sentences', 'Yes']
```

Set 操作





Set 操作

- {} can also be used to create a set
- .add() adds one item
- .update() can add everything from another list

```
>>> myset = set([1,2,3])
>>> myset
{1, 2, 3}
>>> myset.update([4,5,6])
>>> myset
{1, 2, 3, 4, 5, 6}
>>> myset.add("a")
>>> myset
{1, 2, 3, 4, 5, 6, 'a'}
```

```
>>> myset = set([1,2,3,4,5,6,7])
>>> myset.remove(4)
>>> myset
{1, 2, 3, 5, 6, 7}
>>> myset.difference_update([2,5])
>>> myset
{1, 3, 6, 7}
```

Set 操作

- .union() :Combines sets
- .intersection() :Is items common to both sets
- .symmetric_difference: All the items in the sets and remove the intersection from them.

```
>>> a = set([1,2,3])
>>> b = set([3,4,5])
>>> a.difference(b)
{1, 2}
>>> b.difference(a)
{4, 5}
```

```
>>> a.union(b)
{1, 2, 3, 4, 5}
>>> a.intersection(b)
{3}
>>> a.symmetric_difference(b)
{1, 2, 4, 5}
```

Set 操作

- ^: symmetric_difference
- |: union
- -: difference
- &: intersection

```
>>> a = set([1,2,3])
>>> b = set([3,4,5])
>>> a^b
{1, 2, 4, 5}
>>> a|b
{1, 2, 3, 4, 5}
>>> a-b
{1, 2}
>>> a & b
{3}
```

```
>>> a = set([1,2,3])
>>> b = set([3,4,5])
>>> a.difference(b)
{1, 2}
>>> b.difference(a)
{4, 5}
```

```
>>> a.union(b)
{1, 2, 3, 4, 5}
>>> a.intersection(b)
{3}
>>> a.symmetric_difference(b)
{1, 2, 4, 5}
```

網路封包分析

- Scapy 是一款由 python 撰寫的網路封包分析模組
- 像 Wireshark 一樣
- 有嗅探 (Sniffer) 功能，也可以下條件式 (BPF) 過濾封包

```
$ python
>>> from scapy.all import *
```

Scapy

- rdpcap(filename) will read a file containing pcaps into a scapy.PcaketLIst Data structure
  - packetlist=rdpcap(“example.pcap”)
- wrpcap(filename,packrtlist) will write a PacketList to a pcap file
  - wrpcap(“newpacketcpture.pcap”,PacketList2write)
- sniff() can also be used to capture live packets

Scapy

- Use sniff() to capture all packets matching BPF(Berkeley Packet Filter) and pass them to function analyze()
  - sniff(iface=”eth0”,filter=”HOST 10.10.10.10”,store=0,prn=analyze)
- Use sniff() to capture 100 packets matching the BPF(Berkeley Packet Filter)
  - sniff(iface=”eth0”,filter=”SRC HOST 10.10.10.10”,count=100)
- Use sniff() to read a pcap and apply a BPF(Berkeley Packet Filter)
  - sniff(offline=”sample.pcap”,filter=”TCP PORT 80”)

Scapy

- “scapy.PcaketLIst”: The sniff() and rdpcap() functions return this type variable

```
>>> from scapy.all import*
>>> packlist=rdpcap("test.pcap")
>>> packlist.__class__
<class 'scapy.plist.PacketList'>
>>> type(packlist)
<class 'scapy.plist.PacketList'>
>>> dir(packlist)
['__add__', '__class__', '__delattr__', '__doc__', '__format__', '__getattr__',
'__getattribute__', '__getitem__', '__getslice__', '__hash__', '__init__', '__le
n__', '__module__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__set
attr__', '__sizeof__', '__slots__', '__str__', '__subclasshook__', '_dump_docume
nt', '_elt2pkt', '_elt2show', '_elt2sum', 'afterglow', 'conversations', 'diffplo
t', 'display', 'filter', 'hexdump', 'hexraw', 'listname', 'make_lined_table', 'm
ake_table', 'make_tex_table', 'multiplot', 'nsummary', 'nzpadding', 'padding', '
pdfdump', 'plot', 'psdump', 'rawhexdump', 'replace', 'res', 'sessions', 'show',
'sr', 'stats', 'summary', 'timeskew_graph']
```

Scapy

- sniff()`s callback function

```
>>> def my_packet_processor(packetin):
...     print("I got a packet from ", packetin[IP].src)
...
>>> sniff(iface="lo",prn=my_packet_processor,store=0)
('I got a packet from ', '127.0.0.1')
('I got a packet from ', '127.0.0.1')
('I got a packet from ', '127.0.0.1')
('I got a packet from ', '127.0.0.1')
('I got a packet from ', '127.0.0.1')
('I got a packet from ', '127.0.0.1')
('I got a packet from ', '127.0.0.1')
('I got a packet from ', '127.0.0.1')
```

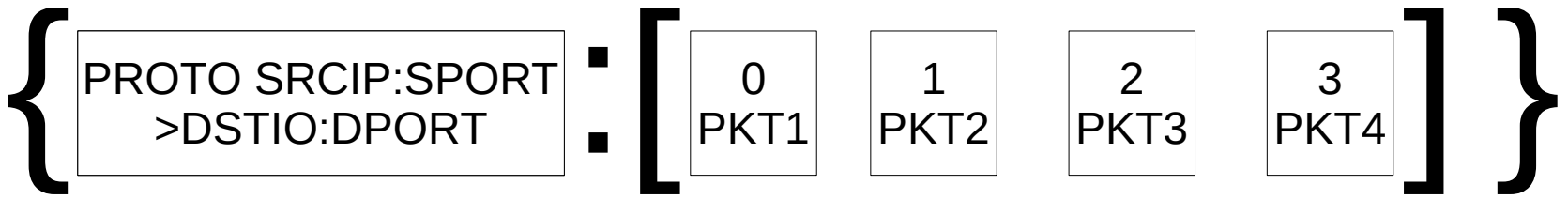
Scapy

- Follow Streams

```
>>> from scapy.all import*
>>> packlist=rdpcap("test.pcap")
>>> len(packlist)
380
>>> packlist[0]
<Ether  dst=00:1d:aa:9f:d1:bc src=34:e1:2d:e9:8d:38 type=0x800 |<IP  version=4 ihl=5 tos=0x0 le
n=771 id=30481 flags=DF frag=0 ttl=64 proto=tcp chksum=0x5dfc src=172.16.105.143 dst=172.217.16
0.110 options=[] |<TCP  sport=36474 dport=https seq=3930623195 ack=3749111478 dataofs=8 reserve
d=0 flags=PA window=1444 chksum=0x6b99 urgptr=0 options=[('NOP', None), ('NOP', None), ('Timest
amp', (971472821, 371982331))] |<Raw  load='\x17\x03\x03\x02\xca\x00\x00\x00\x00\x00\x00\x01\x9
cC\x19\x86\xa1H\xb9\xf8m]9QS\x01)\x16X\x7f\xb80\x9e\x813\xd3\xe6\xae]\xab\xc2&\xa3c\x1e\x87\\*}
H\xe2I3k\x11\xf7\x08*UYq\xd7\xae\xdf\xc8\xa2\xbb\xf9\xfe\xf1\xa9<\xf1\x17\x06\x01:f\xd92\x082\'
\x8c\xe5(\xe2\n\xa4(\xf5\xd5B\x9e\xf1s\xf6>\xe4\xbc)(g_\xb5t\xbc\xb9\x19\xcam$\xa8\x8d\xffx-.\x
d5p\x11M\xbeIm\xf5^1\x96B\xc44x\xf9\n\xf6\xf7G\xeb\xe3\x1b\xa7\xb9\xf6\xc1\xa2J\x06\xf9\xf1\x9a
\xbd\x93{\x95<\xf8\x93\xdf\x9d\x92Ap\xcc\xceef7\xee\x8e6;\x88]\x08\x18z\xbc\xc1\xc5%\t-V\xb7\x0
b\x08S\xe6\x07B\xc2\xb5\xb2\xb1\x11\xebE#\xb5\xcc\x04\x0cK\x9d\xef\xf5\x96\x8a\x8c\xaa\x19\xd7\
x05\xc6\xc4\xe6\x80p\x05%T\xaa\x8e3j4-yag\xd3U\x06\x98\x0e5\xe7\xb6\xb6\xb6\x7fT\xd8\x1e\xc5H\x
a7:*\x9dI\x9d\x97\xcc\xfb\xcf\x1c\xc6\xe55\x07\x04\x1ex(\x83\xbb\xd1\xfc\x18\x92&\xf60\x1d\x94\
x16\xa2\x8b\x08\x9a\x8d\xc6\xae\x90\x7f\x05\x06\xeb=\x04\xfbz\xfc\x90&@\x03\x8d\x1c\xfe}]\x06\x
7f\x97\x96\xb1\xb8b>\x8ewg`Ef5;u\xe6I?\x96\xbd\x10\xc6\xe7!\xabT\xb9\x1feV\xc7yv\xb4\xd4(\x16#r
```

Scapy

- .Sessions()return a dictionary of streams




```
>>> packlist.sessions()
{'TCP 108.177.97.188:5228 > 172.16.105.143:47248': <PacketList: TCP:1 UDP:0 ICMP:0 Other:0>, 'U
DP 172.16.105.130:5353 > 224.0.0.251:5353': <PacketList: TCP:0 UDP:3 ICMP:0 Other:0>, 'TCP 172.
16.105.143:47248 > 108.177.97.188:5228': <PacketList: TCP:1 UDP:0 ICMP:0 Other:0>, 'UDP fe80::1
861:f2fe:693e:7cfb:5353 > ff02::fb:5353': <PacketList: TCP:0 UDP:3 ICMP:0 Other:0>, 'TCP 173.19
4.51.235:443 > 172.16.105.143:53714': <PacketList: TCP:233 UDP:0 ICMP:0 Other:0>, 'TCP 172.217.
160.110:443 > 172.16.105.143:36474': <PacketList: TCP:7 UDP:0 ICMP:0 Other:0>, 'TCP 172.16.105.
143:36474 > 172.217.160.110:443': <PacketList: TCP:4 UDP:0 ICMP:0 Other:0>, 'TCP 172.16.105.143
:53714 > 173.194.51.235:443': <PacketList: TCP:127 UDP:0 ICMP:0 Other:0>, 'IP 172.16.105.130 >
224.0.0.251 proto=igmp': <PacketList: TCP:0 UDP:0 ICMP:0 Other:1>}
>>>
```

Scapy

- .sessions().keys()=A list of strings

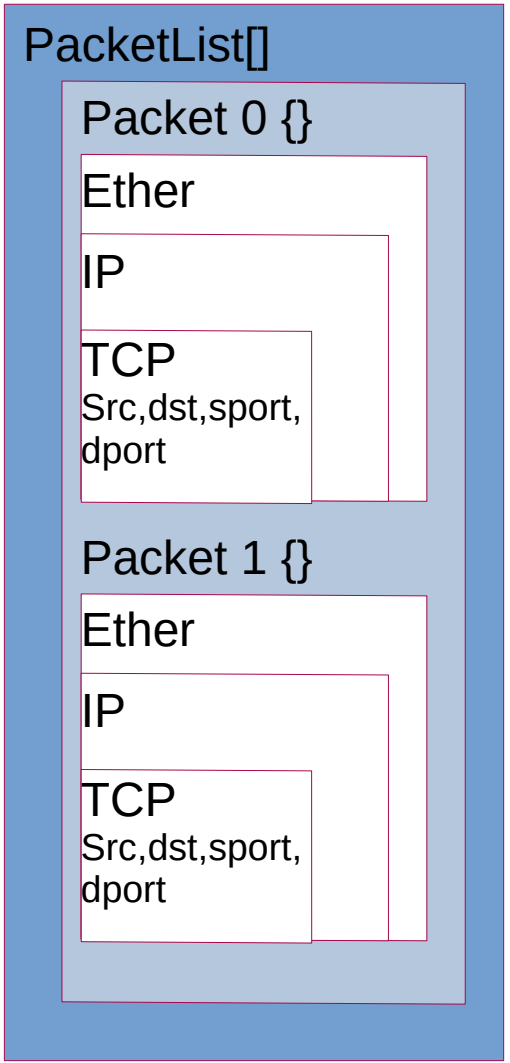
```
>>> from scapy.all import*
>>> pcaklist=rdpcap("test.pcap")
>>> pcaklist.sessions().keys()
['TCP 108.177.97.188:5228 > 172.16.105.143:47248', 'UDP 172.16.105.130:5353 > 22
4.0.0.251:5353', 'TCP 172.16.105.143:47248 > 108.177.97.188:5228', 'UDP fe80::18
61:f2fe:693e:7cfb:5353 > ff02::fb:5353', 'TCP 173.194.51.235:443 > 172.16.105.14
3:53714', 'TCP 172.217.160.110:443 > 172.16.105.143:36474', 'TCP 172.16.105.143:
36474 > 172.217.160.110:443', 'TCP 172.16.105.143:53714 > 173.194.51.235:443', '
IP 172.16.105.130 > 224.0.0.251 proto=igmp']
```

.sessions().values()=A list of PacketLists

```
>>> pcaklist.sessions().values()
[<PacketList: TCP:1 UDP:0 ICMP:0 Other:0>, <PacketList: TCP:0 UDP:3 ICMP:0 Other
:0>, <PacketList: TCP:1 UDP:0 ICMP:0 Other:0>, <PacketList: TCP:0 UDP:3 ICMP:0 O
ther:0>, <PacketList: TCP:233 UDP:0 ICMP:0 Other:0>, <PacketList: TCP:7 UDP:0 IC
MP:0 Other:0>, <PacketList: TCP:4 UDP:0 ICMP:0 Other:0>, <PacketList: TCP:127 UD
P:0 ICMP:0 Other:0>, <PacketList: TCP:0 UDP:0 ICMP:0 Other:1>]
```

Scapy

- PacketList Data Structure
- A PacketList contains one or more packet, similar to a list
- Packets contain one or more layers, similar to “nested” dictionary
- Layers have attributes, similar to an object



Scapy

- PacketLists Have Packets, Packets Have Layers
- Each of packet layers displayed can be addressed by treating the name of the layer as an index. Its value includes the layer and sublayers.
- PacketList[<packet number>][<layer name>]

```
>>> pcaklist[2][TCP]
<TCP  sport=https dport=36474 seq=3749111478 ack=3930623914 dataofs=8 reserved=0
 flags=A window=1050 chksum=0x25ae urgptr=0 options=[('NOP', None), ('NOP', None
), ('Timestamp', (372001262, 971472821))] |>
>>>
```

- You can determine if your packet has a layer with .haslayer(layer)

```
>>> pcaklist[2]
<Ether  dst=34:e1:2d:e9:8d:38 src=00:1d:aa:9f:d1:bc type=0x800 |<IP  version=4 i
hl=5 tos=0x0 len=52 id=46550 flags= frag=0 ttl=58 proto=tcp chksum=0x6806 src=17
2.217.160.110 dst=172.16.105.143 options=[] |<TCP  sport=https dport=36474 seq=3
749111478 ack=3930623914 dataofs=8 reserved=0 flags=A window=1050 chksum=0x25ae
urgptr=0 options=[('NOP', None), ('NOP', None), ('Timestamp', (372001262, 971472
821))] |>>>
>>> pcaklist[2].haslayer(TCP)
True
>>> pcaklist[2].haslayer(UDP)
0
```

Scapy

- PacketList[<packet number>][<Layer name>].<Field name>

```
>>> pcaklist[2]
<Ether  dst=34:e1:2d:e9:8d:38 src=00:1d:aa:9f:d1:bc type=0x800 |<IP  version=4 i
hl=5 tos=0x0 len=52 id=46550 flags= frag=0 ttl=58 proto=tcp chksum=0x6806 src=17
2.217.160.110 dst=172.16.105.143 options=[] |<TCP  sport=https dport=36474 seq=3
749111478 ack=3930623914 dataofs=8 reserved=0 flags=A window=1050 chksum=0x25ae
urgptr=0 options=[('NOP', None), ('NOP', None), ('Timestamp', (372001262, 971472
821))] |>>>
>>> pcaklist[2].haslayer(TCP)
True
>>> pcaklist[2].haslayer(UDP)
0
```

```
>>> pcaklist[2][Ether]
<Ether  dst=34:e1:2d:e9:8d:38 src=00:1d:aa:9f:d1:bc type=0x800 |<IP  version=4 i
hl=5 tos=0x0 len=52 id=46550 flags= frag=0 ttl=58 proto=tcp chksum=0x6806 src=17
2.217.160.110 dst=172.16.105.143 options=[] |<TCP  sport=https dport=36474 seq=3
749111478 ack=3930623914 dataofs=8 reserved=0 flags=A window=1050 chksum=0x25ae
urgptr=0 options=[('NOP', None), ('NOP', None), ('Timestamp', (372001262, 971472
821))] |>>>
>>> pcaklist[2][Ether].dst
'34:e1:2d:e9:8d:38'
>>>
```

Thank you for your attention