py0f

API Documentation

September 17, 2006

Contents

1 Package py0f

py0f - p0f statistics tool v0.1.0 - 17/09/2006

author: jean-michel dot philippe at libertysurf.fr home page: http://philipjm.free.fr/blog/index.php?2006/09/16/48-py0f-010-first-release

1.1 Modules

1.2 Variables

Name	Description
__author__	**Value:** `'jean-michel dot philippe at libertysurf.fr'` *(type=`str`)*
__date__	**Value:** `'17/09/2006'` *(type=`str`)*
__url__	**Value:** `'http://philipjm.free.fr/blog/index.php?2006/09-/16/48-py0f-010-first-release'` *(type=`str`)*
__version__	**Value:** `'0.1.0'` *(type=`str`)*

2 Module py0f.classes

2.1 Variables

Name	Description
AttemptExpr	**Value:** `<_sre.SRE_Pattern object at 0x8417c40>` *(type=`SRE_Pattern`)*
OneDay	**Value:** `0.99999999906867743` *(type=`float`)*
OneHour	**Value:** `0.041666666627861559` *(type=`float`)*
OneMinute	**Value:** `0.00069444444379769261` *(type=`float`)*
OneSecond	**Value:** `1.1574074063294876e-05` *(type=`float`)*
verbose	**Value:** `False` *(type=`bool`)*

2.2 Class Attempts

Class collecting all network connection attempts recorded into p0f logs. Logs are imported with the 'load' method. Each attempt is recorded in a series of lists for IP's, ports, etc. You can then perform some statistics on IP's, ports, etc.

Properties:

- fromIP = full list of connection attempt origin IP's
- fromPort = full list of connection attempt origin ports
- toIP = full list of connection attempt destination IP's
- toPort = full list of connection attempt destination ports
- Date = full list of connection attempt dates
- OS = full list of connection attempt origin OS
- periods = list of log file date ranges
- datemin = overall minimum date (statistics start date)
- datemax = overall maximum date (statistics end date)
- density = list of attempts number per hour for each log file
- rejected = list of rejected record number in each log file
- reject_from_ip = filters origin IP's of this list
- reject_from_port = filters origin ports of this list
- reject_to_ip = filters destination IP's of this list
- reject_to_port = filters destination ports of this list
- reject_os = filters origin OS's of this list
- threshold = don't display statistics below this value (%) except OS's
- maskIP = don't display full IP's (boolean)
- maskUnknown = don't display unknown port services (boolean)

2.2.1 Methods

__init__(*self*, *filename*=None)

Creates an Attempts instance. Eventually loads attempts from a given log file.
```
>>> Logs = Attempts(filename=None)
```
Options:
- filename = name of a log file to be loaded

__getitem__(*self*, *k*)

Returns the n-th attempt as a list.
```
>>> Date, fromIP, fromPort, toIP, toPort = Logs[0]
```

__getslice__(*self*, *k*, *l*)

Returns a subset of attempts as a list of lists Date, fromIP, fromPort, toIP, toPort
```
>>> for Date, fromIP, fromPort, toIP, toPort in Logs[:10]:
>>>   ...
```

__len__(*self*)

Returns the total number of attempts.
```
>>> print len(Logs)
```

__str__(*self*)

Displays statistics on attempts. Some instance properties control the text output:
- threshold = don't display statistics below this value (%) except OS's
- maskIP = don't display full IP's (boolean)
- maskUnknown = don't display unknown port services (boolean)

```
>>> print Logs
```

filter(*self*, *index*, *value*)

Returns a subset of attempts obtained by filtering all attempts.
```
>>> AttemptList = Logs.filter(index, value)
```
Input:
- index = index of the data which is tested in the list returned by Logs.__getitem__, ie. [Date, fromIP, fromPort, toIP, toPort]
- value = value to match to pass test

Output:
- AttemptList = list of filtered attemps as lists [Date, fromIP, fromPort, toIP, toPort]

from_ip_stats(*self*, *from_date*=None, *to_date*=None, *date_string*='%a %b %d %H:%M:%S %Y')

Computes IP origin statistics. You can specify a date range of analysis.
>>> StatDict = Logs.from_ip_stats(from_date=None, to_date=None, date_string=DateFormat)
Options:

- from_date = the start date as a string or a number (None = overall start)
- to_date = the end date as a string or a number (None = overall end)
- date_string = the date format string

Output:

- StatDict = dictionary of IP origin occurrences

from_port_stats(*self*, *from_date*=None, *to_date*=None, *date_string*='%a %b %d %H:%M:%S %Y')

Computes port origin statistics. You can specify a date range of analysis.
>>> StatDict = Logs.from_port_stats(from_date=None, to_date=None, date_string=DateFormat)
Options:

- from_date = the start date as a string or a number (None = overall start)
- to_date = the end date as a string or a number (None = overall end)
- date_string = the date format string

Output:

- StatDict = dictionary of port origin occurrences

load(*self*, *filename*)

Loads attemps from a p0f log file. Rejected attemps are not recorded. Also computes the log file date range, density, number of rejected attempts and the overall date limits.
`>>> Logs.load(filename)`
Input:
- filename = name of the log file

Attempt information are stored into the following properties:
- fromIP = full list of connection attempt origin IP's
- fromPort = full list of connection attempt origin ports
- toIP = full list of connection attempt destination IP's
- toPort = full list of connection attempt destination ports
- Date = full list of connection attempt dates
- OS = full list of connection attempt origin OS
- periods = list of log file date ranges
- datemin = overall minimum date (statistics start date)
- datemax = overall maximum date (statistics end date)
- density = list of attempts number per hour for each log file
- rejected = list of rejected record number in each log file

Attempt rejected rules is defined by the following properties:
- reject_from_ip = filters origin IP's of this list
- reject_from_port = filters origin ports of this list
- reject_to_ip = filters destination IP's of this list
- reject_to_port = filters destination ports of this list
- reject_os = filters origin OS's of this list

loadFiles(*self*, *FileList*)

Loads a series of p0f log files into an Attempts instance.
`>>> Logs.loadFiles(FileList)`
Input:
- FileList = list of log files to be loaded

os_stats(*self*, *from_date*=`None`, *to_date*=`None`, *date_string*=`'%a %b %d %H:%M:%S %Y'`)

Computes OS origin statistics. You can specify a date range of analysis.
`>>> StatDict = Logs.os_stats(from_date=None, to_date=None, date_string=DateFormat)`
Options:
- from_date = the start date as a string or a number (None = overall start)
- to_date = the end date as a string or a number (None = overall end)
- date_string = the date format string

Output:
- StatDict = dictionary of OS occurrences

to_ip_stats(*self*, *from_date*=None, *to_date*=None, *date_string*='%a %b %d %H:%M:%S %Y')

Computes IP destination statistics. You can specify a date range of analysis.
>>> `StatDict = Logs.to_ip_stats(from_date=None, to_date=None, date_string=DateFormat)`
Options:

- from_date = the start date as a string or a number (None = overall start)
- to_date = the end date as a string or a number (None = overall end)
- date_string = the date format string

Output:

- StatDict = dictionary of IP occurrences

to_port_stats(*self*, *from_date*=None, *to_date*=None, *date_string*='%a %b %d %H:%M:%S %Y')

Computes port destination statistics. You can specify a date range of analysis.
>>> `StatDict = Logs.to_port_stats(from_date=None, to_date=None, date_string=DateFormat)`
Options:

- from_date = the start date as a string or a number (None = overall start)
- to_date = the end date as a string or a number (None = overall end)
- date_string = the date format string

Output:

- StatDict = dictionary of port occurrences

3 Module py0f.filter

3.1 Variables

Name	Description
SenderExpr	**Value:** <_sre.SRE_Pattern object at 0x841e2a0> *(type=SRE_Pattern)*

4 Module py0f.graphics

4.1 Functions

pie(*DataDict*, *threshold*=`1.0`, *translatefunc*=`None`, *loc*=`0`)

Draws pie graphics for OS origin and port destination statistics.
`>>> pie(DataDict, threshold=1.0, translatefunc=None, loc=0)`
Input:

- DataDict = dictionary of data value occurrence

Options:

- threshold = don't display occurrences below this threshold (%)
- translatefunc = a function which translates dictionary keys into more readable information, eg. ports numbers into service names
- loc = legend location (argument of pylab.legend)

Example:

```
>>> pie(Logs.to_port_stats(), translatefunc=lambda x: py0f.utils.getService(x)[0]+" (%s)"%x)
The translation function returns "ssh (22)" for an input value of 22.
```

5 Module py0f.utils

5.1 Functions

_(*Text*)

in prevision of a future internationalisation?

```
>>> Text = _(Text)
```

date2num(*date*, *date_string*='%a %b %d %H:%M:%S %Y')

```
Computes a double date value from a date string. Returned values
are compatible with the matplotlib date representation for graphics.

>>> x = date2num(dates, date_string=DateFormat)

Input:
        - date                  = date string to be converted to double
Options:
        - date_string   = date format of input date strings
Output:
        - x                             = date value

Note:
        Returned numbers are floating point numbers which give
        number of days since 0001-01-01 00:00:00 UTC (fraction
        part represents hours, minutes, seconds)
```

getFileList(*arglist*, *filefilter*)

Returns a file list from a list of files or directories, given a file filter.

```
>>> FileList = getFileList(arglist, filefilter)
```

Input:

- arglist = list of files or directories
- filefilter = file filter, eg. '*.log'

Output:

- FileList = list of target files

getService(*Port*)

Returns the port number as a string and its description when available. Uses global variables:

- ServiceList = list of strings extracted from a /etc/services-like file
- ServiceMatch = regular expression matching a port description line

```
>>> PortStr, PortDescr = getService(Port)
```

Input:

- Port = the port number (integer)

Output:

- PortStr = the port number as a string
- PortDescr = the port description, "no description available" when no description was available

num2date(*value*, *date_string*='%a %b %d %H:%M:%S %Y', *tz*=None)

```
Computes a date string from a date value. Input values must be
compatible with the matplotlib date representation for graphics.

>>> dates = num2date(values, date_string=DateFormat, tz=None)

Input:
        - value                 = date value
Options:
        - date_string   = date format of output date strings
        - tz                    = datetime module timezone instance
Output:
        - date                  = date string

Note:
        Input numbers are floating point numbers which give
        number of days since 0001-01-01 00:00:00 UTC (fraction
        part represents hours, minutes, seconds)
```

sortdict(*Dict*)

Sorts a dictionnary on its values.

```
>>> Pairs = sortdict(Dict)
```

Input:

- Dict = the dictionnary to be sorted

Output:

- Pairs = list of sorted pairs (key, value)

storeServices(*File*, *Filter*)

Returns a cleaned version of a /etc/services-like file for getService.
>>> `List = storeServices(File, Filter)`
Input:
- File = file to be loaded and cleaned
- Filter = regular expression instance for file lines filtering

Output:
- List = list of selected file lines (strings)

str2list(*Text*)

Convert a comma separated list into a list of strings.
>>> `List = str2list(Text)`
Input:
- Text = the comma separated list (string)

Output:
- List = list of strings

5.2 Variables

Name	Description
DateFormat	**Value:** `'%a %b %d %H:%M:%S %Y'` *(type=str)*
ServiceFilter	**Value:** `<_sre.SRE_Pattern object at 0xb6db6bf0>` *(type=SRE_Pattern)*
ServiceList	**Value:** `['tcpmux\t\t1/tcp\t\t\t\t# TCP port service multiplexer\n', 'echo\t\t7/tcp\n'...` *(type=list)*
ServiceMatch	**Value:** `'([\\w-]+)\\s+%d/\\S+\\s+(.+)'` *(type=str)*
ServicesFile	**Value:** `'/home/jm/dev/py0f/services'` *(type=str)*

Index