USEFUL PYTHON LIBRARIES FOR CREATING IDS SOFTWARE

Ivan Babić[1]
Aleksa Maksimović
Slobodan Nedeljković
Ministry of the Interior of the Republic of Serbia

Mihailo Jovanović
Office for IT and eGovernment, Serbia

Milan Čabarkapa, PhD
School of Electrical Engineering, University of Belgrade, Serbia

Dragan Randjelović, PhD
Full professor in retirement, University of Criminal Investigation and Police Studies, Belgrade, Serbia

Abstract: This paper considers contours of the possibilities of applying the selected Python libraries in order to detect intrusion attacks whose purpose is to penetrate secured network or secured computer. This paper elaborates Python libraries that can be used for monitoring network traffic and eventually use them for making IDS software (Intrusion detection system). After preview of libraries with those capabilities, a very simple use case of IDS software solution is going to be presented, so that the readers can see how the mentioned libraries can be used for that kind of software.

Keywords: Python, IDS, Eddie, Pypcap, Scapy, SimpleMonitor

INTRODUCTION

Nowadays, IT industry is facing one of its biggest challenges and that is computer networks security. We are living in a time where DDoS (Distributed Denial of service) attacks are present everyday and they are available for anyone who is willing to pay certain amount of money, since even with some basic computer knowledge one man can do it and make some serious damage to the victim.

[1] bicba90@gmail.com

That is the reason why prevention and short time respond to these attacks is crucial, and eventually one of the reasons why this paper is written. IDS systems are playing a big role in detecting those attacks, it does not matter if it is hardware of software system. This paper is focusing on software IDS systems and using existing Python libraries is going to represent how effectively malicious attack can be detected and separated from legitimate network traffic. IDS and Firewall is not the same thing, that is because IDS does not prevent attack, but instead of that IDS alarms user that attack is happening or has already happened. In order to achieve that, we need a few libraries to make them work together, so that the alarm triggers at right time. Libraries that are represented in this paper are Eddie – used for monitoring network and file system, Pypcap – used for recording incoming packages, LinkChecker - used for link checking and detailed website analysis with possibility of creating website map, Scapy – used for recording and decoding network packages, protocols and request that are received and Simple-Monitor library that is used for monitoring hosts and network connections. Using some of the mentioned libraries we are going to make application with purpose to detect attacks and an alarm user about that. Other programming languages are also used for this purpose, for example JAVA (Northcutt & Novak, 2003) and C++ like standards for Web and back-end programming, also Ruby is used because it is allround language just like Python that are discussed in this paper.

LIBRARIES PREVIEW

Tools

As we have said before, Python programming language is used in this paper. We have chosen Python because it is a script language and it is very flexible. A more prevalent version of Python in use is version 3. Developer environment that is chosen is PyCharm (Kroger, 2015) because of its simple IDE interface that is well known and for its numerous options such as, debugging, marking syntax and other errors, simple project structure and lots of other options.

Eddie

Eddie is also called monitoring agent. It can be used as a standalone tool and perform actions, checks and tests that are predefined. Eddie has usage in network monitoring, file system check, HTTP checks, POP3 tests, SNMP queries, RADIUS authentication tests, process tracking, network configuration, ping check, TCP ports check, monitor file changes and scanning file logs (The EDDIE Tool project, 2008). This library is interesting for us because of its network monitoring and gathering data capabilities, for what we usually need several libraries and here it is all in one place. Whole library is developed in 100% Python and it can be started on any computer with running Python environment. Rules of monitoring are creating in a similar way as Python regular expressions and they can go from the simplest one to the most complicated one, depending on what we need.

Pypcap

This library has wide application in creating tools that are used for network security, precisely in our paper it is very useful for detecting network attacks. This library, beside its other functions, serves for monitoring network traffic, more specific for recording and transfer of network packages, and it can also be used for filtering packages at preprocessor level and for network statistics (Pypcap, 2018).

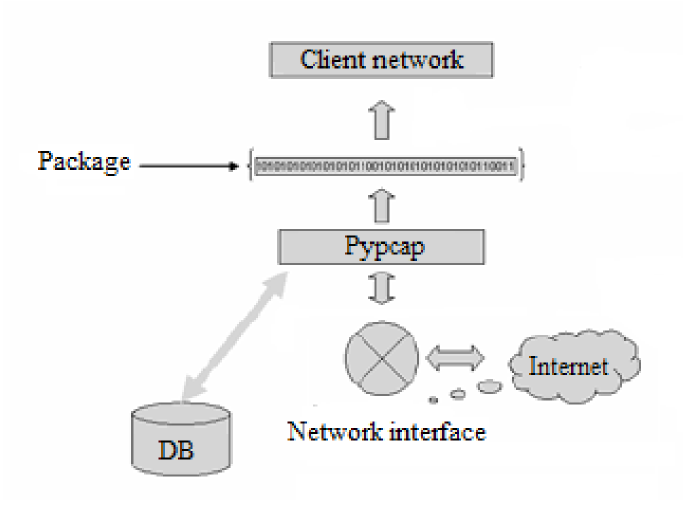
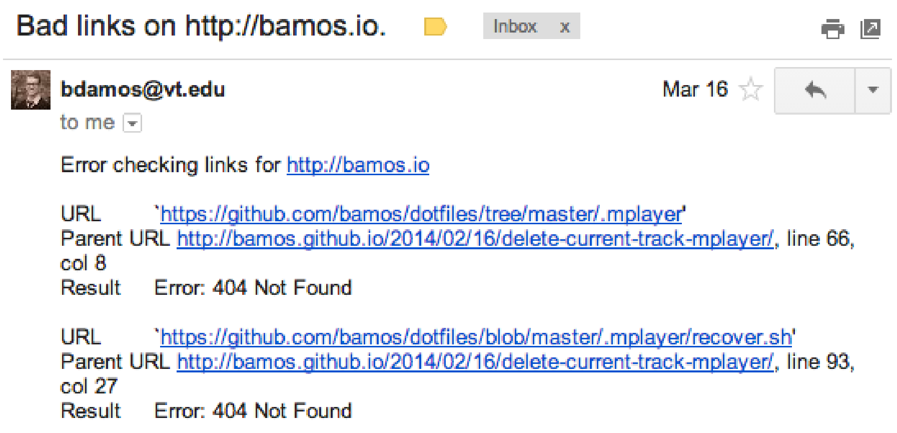


Figure 1. *Storing an incoming package in DB*

The example shown in Figure 1 clearly shows how Pypcap would store a package and package information into the database for further analysis. It is important to say that we do not intercept the package, but only collect information about it, because we are making a software for detecting an intrusion, and not for blocking it. The most important information for us is the number and size of incoming packages in a period of time. Analyzing those data, we can detect DDoS attack that we have mentioned before.

LinkChecker

As its name says, this library is useful for link checking, or in other words for website validation. Links can be checked separately or whole web pages at once (LinkChecker, 2008). This library returns data on web page availability, page name, page response time, page size, page last modification date, and on errors on that page. A few more things can be obtained using this library, such as the number of images on a specific web page, the number of applications, text content, video, mail addresses etc… those things are not so important as that we have mentioned before, because with those previous we can analyze server pages in details. This library is the most useful for extracting data out of websites because output data can be shown in few structured formats such as HTML, XML, CSV, SQL or simply as a graphic representation of website map. All these structured formats are easy to work with and process them later.



Bad links on http://bamos.io.

bdamos@vt.edu

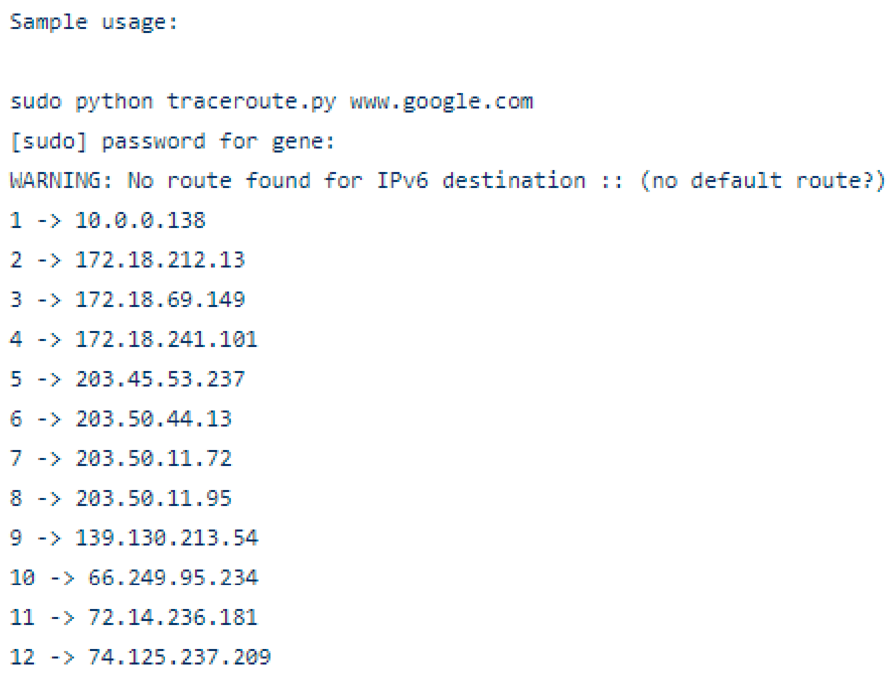
```
Error checking links for http://bamos.io

URL        `https://github.com/bamos/dotfiles/tree/master/.mplayer'
Parent URL http://bamos.github.io/2014/02/16/delete-current-track-mplayer/, line 66,
col 8
Result     Error: 404 Not Found

URL        `https://github.com/bamos/dotfiles/blob/master/.mplayer/recover.sh'
Parent URL http://bamos.github.io/2014/02/16/delete-current-track-mplayer/, line 93,
col 27
Result     Error: 404 Not Found
```

Figure 2. *LinkChecker result*

Figure 2 shows the result of LinkChecker webpage validation, where the result is showing unavailable links on the requested webpage. This library may not have its application in network attacks detection, but it can provide us with useful data that we get after analyzing data of the attack which has previously happened.

Scapy

Scapy library has some possibilities that other libraries do not. It can decode various number of different protocols network packages to send them, store them, make a request and response comparison and a lot more. It is a very flexible library, easy to combine with other tools. Any functionality that is integrated can be used with existing libraries (Scapy project, 2009). As it is, it represents a perfect choice for making a new tool. As we have said before, it can be used for sending packages, which is important part in making IDS software, because by sending packages we can test its functionality and efficiency. The best option of Scapy library is the so-called Traceroute that is tracing a route of a network package. Any package poses a 8 bit TTL field that is decremented every time it passes over some router on its way to destination, the feature that is implemented to prevent infinite package circulation over internet or network. When TTL value hits zero, the ICMP error comes out, which represents that the package time has expired, that response has been returned to the sender of the package. Scapy uses that fact to find all jumps, basically IP addresses of the router that package passed over on its way. This is very useful feature that can help in finding the attacker.



```
Sample usage:


sudo python traceroute.py www.google.com
[sudo] password for gene:
WARNING: No route found for IPv6 destination :: (no default route?)
1 -> 10.0.0.138
2 -> 172.18.212.13
3 -> 172.18.69.149
4 -> 172.18.241.101
5 -> 203.45.53.237
6 -> 203.50.44.13
7 -> 203.50.11.72
8 -> 203.50.11.95
9 -> 139.130.213.54
10 -> 66.249.95.234
11 -> 72.14.236.181
12 -> 74.125.237.209
```

Figure 3. *All jump to webpage www.google.com*

Figure 3 represents all jumps to the webpage www.google.com. It is easily seen that 12 jumps have been made and IP address of every router over that package travelled to its destination. From the above it is clearly seen why this library is very important from the perspective of network security and finding a source of an attack.

SimpleMonitor

SimpleMonitor is Python script that tracks hosts and network connection. It is made in order to be fast and easy to use, to be more convenient for smaller companies and home application. Instances of this script can send results to one centralized location (Simplemonitor project, 2015).

Some of the functionalities that SimpleMonitor supports are:

- Ping monitoring
- TCP monitoring (scanning is there any open ports on host)

- HTTP monitoring (checks whether we can download URL without error and also if the page content responds to our query)
- DNS monitoring
- Services monitoring
- Windows services, daemontools services
- Hard drive free space monitoring
- File creation time
- and a lots of others…

A lot of options can be added to work together with this script and it is fairy simple for anyone who knows how to code in Python

Options of SimpleMonitor for logging and alarming are:

- Storing all tracking processes state and all other iteration into SQLite database
- Storing current tracking processes state into SQLite database
- Sending warning email when error occurs with tracking process and when it is restored to original state, directly over SMTP or Amazon SES
- Writing into log file all successful and unsuccessful made operations, or only unsuccessful
- Sending SMS messages over BulkSMS service (subscription needed)
- Writing HTML status page
- Writing entries into syslog (all except Windows)
- Sending notifications over Slack, 46elks, Notify My Android, Pushbullet and Pushover
- Execution of commands when error occurs with tracking process and when it is restored to original state

Furthermore, options for logging and alarming can be added easily here by writing some more code in Python.

More options that SimpleMonitor supports:

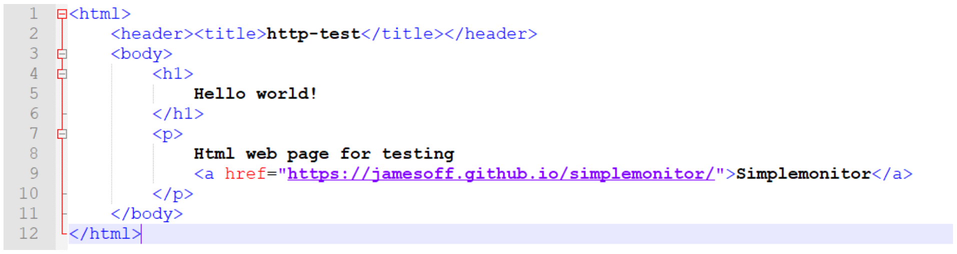
- Simple configuration of file format: standard is INI file for configuration
- Dependency: SimpleMonitor can be adjusted to depend on status of other processes. If process has failed, processes that depends on it will be skipped until that process becomes successful
- Tolerance: Tracking process can be set to alarm after certain number of failure attempts
- Alarm escalation: Alarms can be set so that after a failure occurs a few time in a row (after tolerance limit excided) before its get triggered, so it can be sent to more addresses
- Urgency: Processes can be set as non-urgent so we do not use some of the alarming options on them such as SMS

- Process by host: Defying process to track only one host and to ignore others, so that configuration file can share between all hosts
- Process intervals: Processes are set in that way for example to make checks every 60 seconds, but also can be set to make checks once a day
- Alarm periods: Alarms can be set to trigger only at certain time of the day or at specific days only
- Alarm snoozing: For example if process fails during the night, we do not want to be disturbed in the middle of the night by SMS, but at 7 o'clock in the morning we are going to receive a SMS that error has occurred on that process
- Distant processes: Processes that are on distant computers can send their results to centralized instance with purpose of logging and alarming

Each of these options that we have mentioned above and which are part of the SimpleMonitor script can be very useful for creating our IDS software. Options as regards alarming are extremely important and flexible, so they have a wide application for us.

USE CASE OF IDS SOFTWARE

In this use case, one SimpleMonitor will be uploaded in one Linux machine to monitor the operation of a web site hosted on one remote Windows-machine, and in case the page becomes inaccessible, it will send an email to a specific address. In order to be able to perform this experiment, one Windows-machine must be set up to host a web site. To accomplish this, we used the Windows 10 machine with the Wampserver version 3.1.4 software package installed in which Apache 2.4.35 is used to host the web site. After installing Wampserver on the partition C: of this Windows-machine, a folder named wamp64 appeared, including, among other things, a folder www. Within this folder, we created a folder called http-test and placed an index.html file in it shown in Figure 4.



```
<html>
    <header><title>http-test</title></header>
    <body>
        <h1>
            Hello world!
        </h1>
        <p>
            Html web page for testing
            <a href="https://jamesoff.github.io/simplemonitor/">Simplemonitor</a>
        </p>
    </body>
</html>
```

Figure 4. *Content of index.html*

In addition, we need a remote Linux machine. In our case, the Ubuntu 18.04.2 LTS version was used on which the Python 3.6.7 version was installed. Linux and Windows-machines are within the same network.

Within the Linux machine via a search engine, by typing a link to a web page hosted on the web page, this page is displayed in the browser. Within the Linux machine, a new virtual environment for Python is created. We do this by typing the following commands into the terminal:

```
python3 -m venv env
source ./env/bin/activate
```

The SimpleMonitor is downloaded from the https://jamesoff.github.io/simplemonitor/ website. SimpleMonitor is downloaded as a zipped folder which, when extracted, comes as a string of scripts and configuration files. In order to work with these scripts, it is necessary to bring all the libraries into the new virtual environment we created. This is done by opening the terminal to the folder, where SimpleMonitor is located and running the command:

```
pip install -r requirements.txt
```

The next step is to create configuration files. Within the same folder, two files "monitors.ini" and "monitor.ini" are created. Monitors.ini represents a list of elements that are monitored. In this configuration file, the type of monitoring is defined, as well as the required parameter for this type of monitor, which is a link to a web page. In this case, this file is filled using following settings:

```
[website-http]
type=http
url=http://10.233.209.188/http-test/
```

monitor.ini represents a configuration file for how to monitor and how to react in a particular situation. In our case, the configuration file looks like this

```
[monitor]
interval=10

[reporting]
loggers=logfile
alerters=email

[logfile]
```

```
type=logfile
filename=monitor.log
only_failures=1

[email]
type=email
host=smtp.gmail.com
port=587
from=test1bad1usb@gmail.com
to=test@test.com
username=**********
password=*********
ssl=starttls
```

Here, settings are such that the connection will be refreshed every 10 seconds, that error reports will only be written in log files named moniotr.log and that in case of an event registered as an Alert, an email will be sent to a Gmail address defined by another Gmail address using the Gmail address.

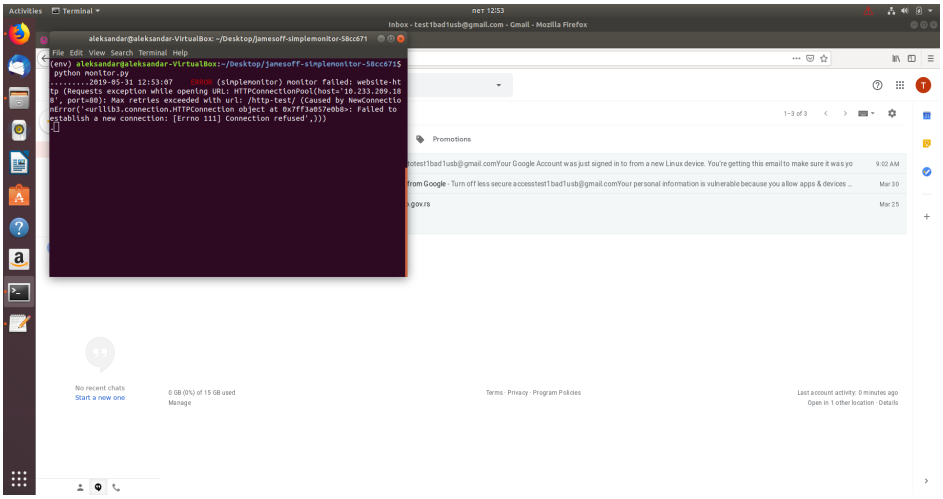
Now that everything is set, we can run the monitor command:

```
python monitor.py
```

The monitor does not display because it successfully communicates with the web page.

If we stop the work of the server, the web page will no longer be available and the monitor will send us the alert that will send the email to the defined address.

At that point, SimpleMonitor realizes that communication with the server is lost and triggers an alert on it.

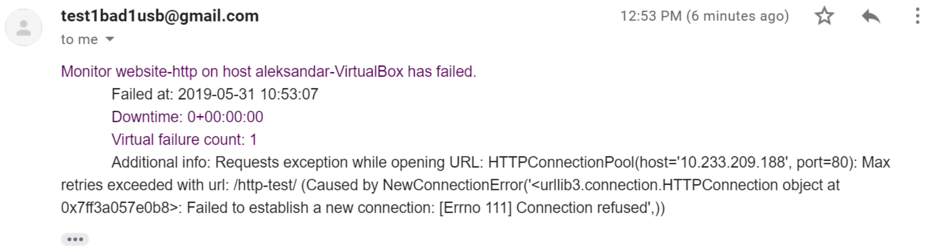


```
aleksandar@aleksandar-VirtualBox: ~/Desktop/jamesoff-simplemonitor-58cc671
(env) aleksandar@aleksandar-VirtualBox:~/Desktop/jamesoff-simplemonitor-58cc671$
 python monitor.py
.........2019-05-31 12:53:07    ERROR (simplemonitor) monitor failed: website-ht
tp (Requests exception while opening URL: HTTPConnectionPool(host='10.233.209.18
8', port=80): Max retries exceeded with url: /http-test/ (Caused by NewConnectio
nError('<urllib3.connection.HTTPConnection object at 0x7ff3a057e0b8>: Failed to
establish a new connection: [Errno 111] Connection refused',)))
.
```

Activities
Terminal
Inbox - test1bad1usb@gmail.com - Gmail - Mozilla Firefox
File Edit View Search Terminal Help
Promotions
totest1bad1usb@gmail.comYour Google Account was just signed in to from a new Linux device. You're getting this email to make sure it was yo 9:02 AM
from Google - Turn off less secure accesstest1bad1usb@gmail.comYour personal information is vulnerable because you allow apps & devices ... Mar 30
.gov.rs Mar 25
No recent chats
Start a new one
0 GB (0%) of 15 GB used
Manage
Terms · Privacy · Program Policies
Last account activity: 0 minutes ago
Open in 1 other location · Details

Figure 5. *Monitor raising error*

As soon as the alarm is raised as we can see in Figure 5, a new e-mail error message shown in Figure 6 is sent to the defined e-mail address



test1bad1usb@gmail.com 12:53 PM (6 minutes ago)
to me

Monitor website-http on host aleksandar-VirtualBox has failed.
Failed at: 2019-05-31 10:53:07
Downtime: 0+00:00:00
Virtual failure count: 1
Additional info: Requests exception while opening URL: HTTPConnectionPool(host='10.233.209.188', port=80): Max retries exceeded with url: /http-test/ (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7ff3a057e0b8>: Failed to establish a new connection: [Errno 111] Connection refused',))

Figure 6. *Received email*

With these settings, the notifications that the server is not available will be sent over and over until the server becomes available again.

In JAVA programming language this solution would be much larger for coding, and more complex itself. For the same thing we would need a few libraries like libraries for packet capturing, data processing, etc. (Ezin, Eugène & Djihountry, Hervé 2011). Additionally, a large number of classes because of the code readability must be separated into smaller parts for easier manipulation, while in case of Python that is not a problem.

CONCLUSION

As we can see, it is fairly easy to start and build a simple IDS software. This is just an example of that, where we can see possibility of alarming through email when webpage becomes unavailable. Even this SimpleMonitor script can help for some basic user needs, such as detecting DDoS attack or server failure, it is really easy to add some features to this for some more information regarding alarming. Furthermore, it is possible to make this work with some other tools that would make it even easier for a user to figure out what is happening and what needs to be done to be protected from an attack, or to fix a problem.

Besides, IDS software can be made in other languages as we have mentioned in the introduction section, like JAVA and C++, but advantages of Python programming language are that Python is a high-level, interpreted and general-purpose dynamic programming language that focuses on code readability. The syntax in Python helps the programmers to do coding in fewer steps as compared to Java or C++. It has large support for libraries, easy integration with other systems, and increased productivity as well. Disadvantages of Python programming language are those that it is weak in mobile computing, difficult for programmers to learn other languages because they are used to extensive libraries and other Python features, it has design restriction and errors that require more testing time than other programming languages, and finally it has primitive database access layer compared with JDBC or ODBC.

REFERENCES

1. Northcutt & Novak (2003) Network Intrusion Detection, New Riders Publishing
2. Kroger, P.K. (2015). Modern Python Development With PyCharm, Pedro Kroger
3. The EDDIE Tool project (2008). Psychofx.com [Weblog]. Retrieved on 20. Jul 2019, from http://eddie-tool.psychofx.com/
4. Pypcap (11. decembar 2018). Readthedocs.org [Weblog]. Retrieved on 20. Jul 2019, from https://buildmedia.readthedocs.org/media/pdf/pypcap/latest/pypcap.pdf
5. LinkChecker (2008). GitHub.com, Retrieved on 20. Jul 2019, from https://github.com/linkchecker/linkchecker
6. Scapy project (2009). Scapy.org, Retrieved on 20. Jul 2019, from https://scapy.org/
7. Simplemonitor project (2015), GitHub.com, Retrieved on 20. Jul 2019, from https://jamesoff.github.io/simplemonitor/
8. Sarker, M.O.F.S. (2014). Python Network Programming Cookbook

9. Ezin, Eugène & Djihountry, Hervé (2011) Java-Based Intrusion Detection System in a Wired Network. International Journal of Computer Science and Information Security. 9. 33-40.