Network Security: ARDUINO Yun Based IDS

Y.SENHAJI[1] , H.MEDROMI[2]

[1] Architecture System Team Hassan II University de Casablanca ENSEM
Casablanca, Morocco

[2] Architecture System Team Hassan II University de Casablanca ENSEM
Casablanca, Morocco

Abstract

This research deals with the issue of computer security, which aims to develop a robust and independent security architecture to reduce the analysis loading and the number of false positives and false negatives. The architecture consists of several probes semantically distributed according to three threat detection methods and presented as an embedded solution on Arduino YUN boards.

Keywords: Network Security, IDPS, Real Time, Embedded System, Distributed System, Arduino.

1.Introduction

Among the major computer security features, we find the firewall systems. However, they do not protect the confidentiality of data. For this purpose, we require the use of cryptographic algorithms to ensure the confidentiality of exchanges and customers.

Moreover, a service based on the IP address to identify its customers can easily be a victim of IP spoofing or ARP spoofing.

As solution of these cases, there is what we call IDPS (Intrusion Detection and Prevention Systems).

IDPS has also the ability to take measures and to detect incidents according to security policy.

However, setting an IDPS is very delicate since a false alarm is a waste of time and money.

This research topic focuses on securing computer networks by using an embedded IDPS on an Arduino Yun Board.

2.The State of the Art

In this section we are going to focus on introducing the concepts of IDPS, distributed system and the Arduino Yun Board. These three points will be the core of our conception development.

2.1 IDPS Definition

Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices. [2]

An intrusion detection system (IDS) is software that automates the intrusion detection process. An intrusion prevention system (IPS) is software that has all the capabilities of an intrusion detection system and can also attempt to stop possible incidents. [2]

Thus, an IDPS is focused on:

Identification of the possible incidents.

Information of access of these incidents.

The blocking of these incidents.

Establishment of the log of these incidents to the security administrators. [2]

Moreover, the IDPS can be exploited for other purposes, such as:

Identifying security policy problems.

Documenting the existing threat to an organization.

Deterring individuals from violating security policies. [2]

The IDPS became a necessary complement to the security infrastructure of each company. They record information about observed events, notify security administrators of the most important of them, and produce reports. They can also evolve in the security environment.

2.2 Common Detection methods

2.2.1 Signature Based Detection

A signature is a model which corresponds to a known threat.

This method is based on the comparison of the units of activities (Package, Log Entry) to the list of these signatures by using the operators of comparison.

However, this method represents two disadvantages:

- It can’t make the link between the request and its response.
- It does not remember the attacks.

2.2.2 Anomaly Based Detection

This method is based on the comparison between the events and the definition of the events considered normal to detect deviations. In this case, the IDPS has a “Profile” which represents the normal behavior. Examples: lists of the users, hosts, connections networks…

It is a method basing itself on statistical calculations (Ex: numbers Email sent by user, a number of tests of erroneous login). So the second disadvantage of the first method is avoided, because the system is fixed with a training phase for generating the profile. This profile could still be regenerated after another measurement of the system. However, if during the generation or regeneration of the profile, the system includes harmful activities, which the rate of change is very small, they will be an integral part of the profile.

2.2.3 Stateful Protocol Analysis

This step of analysis is based on the comparison between the protocols and their profiles. In addition, it exploits the combination of the request and its answer to be able to evaluate the state what constituted a weak point of the first method (Signature based detection).

2.3 Distributed System

A Distributed system can be distributed based on an existing conceptual distance between its components.
This distance can be:

- **Spatial:** distribution by different processes assigned to solve a problem related to space.
- **Semantic:** distribution by the specificity of knowledge and a particular know-how.
- **Structural:** representations are heterogeneous and reasoning mechanisms are different.
- **Semantic:** according to its function and its role within the system.

2.4 Arduino Yun Presentation

The Arduino Yun is an electronic board that uses the Atmel processor ATmega32U4. Besides of that, it has an additional processor: Atheros AR9331, that turn the Linux distribution OpenWrt Linino.

Figure 1 Arduino Yun Board

3.Problematic

To detect and eliminate attacks (security threat), a system must have tools to monitor data in transit. Thus, to analyse all network traffic seems to be the most ideal solution. However, this is an unthinkable alternative because the quantity of data will be very huge and analysis time will be very long (real-time problem). Moreover, even with focusing only on a part of the traffic, load analysis remains important and this analysis must be in a real-time to serve the rapid availability of information. So the question is how to design a security system that can reduce this loading, being autonomous and being indifferent to the vulnerability of the OS of the system to protect.

And of course we should not forget that we are in an academic research, so the solution must be free and open source.

Thus, and in order to respond to these points above, we will try to present in the following section a proposition of an architecture based on the concepts introduced in the state of the art section: Distributed system based on an Arduino Yun Board.

4.Proposed Architecture

4.1 Introduction

Prior to deployment of the security solution, we assume that users are aware of the importance of security and its challenges and that all systems and applications are constantly updated (security patches).

Suppose we have a network with the following elements:

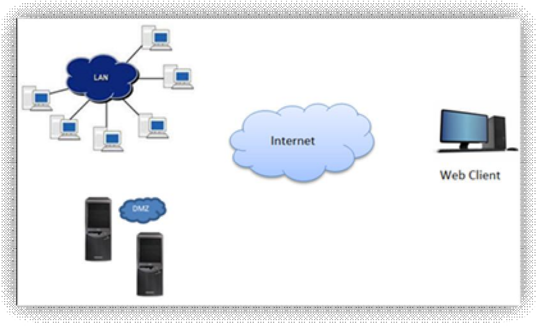


Figure 2 Network Architecture

- A LAN (local area network): consists of several workstations.
- A DMZ (demilitarized zone): Consisting of machines on the internal network that need to be accessible from the outside (mail server, FTP server, web server ...)
- A Web Client: consists of Outside Network

4.2 Semantic Distribution

In this section we are going to operate a Semantic distribution based on IDPS method detection. This distinction aims to specialise the IDPS and specialization causes reducing of loading and of the number of false positives and false negatives. So our IDPS probe will be composed of three Arduino boards that will call ARDS and that we will install in our network:

- ARDS-SPA: Based on the "Stateful Protocol Analysis" as a method of detection
- ARDS-ABD: Based on "Anomaly Based Detection" as a method of detection
- ARDS-SBD: Based on "Signature Based Detection" as a method of detection.

Thus, each ARDS will focus on a detection method to analyse the system.

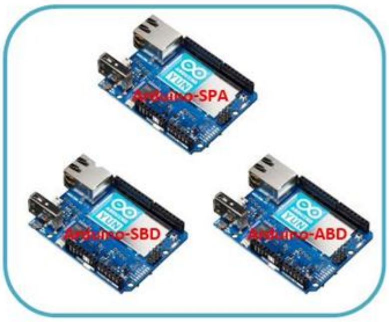


Figure 3 Arduino Semantic distribution

5.PLATFORM DEVELOPMENT

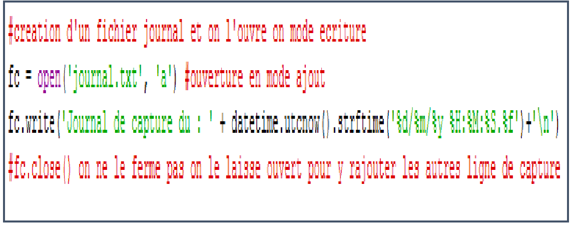
5.1 Capturing traffic

For Traffic capturing we are going to use Raw Socket on Open WRT which is embedded on the Arduino Board. We use a Python script.

Our program has 4 files:

- « regle.txt » that contains security rules
- « journal.txt » for the storage of all captured traffic
- « intrusion.txt » for intrusion logging
- « analyse » for the analysis python script

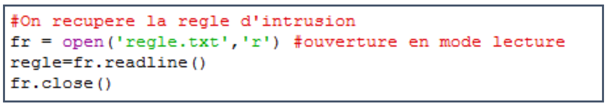
We will be based in the simulation on an intrusion test of a specific IP address. Our analysis script begins with the creation of the log file that will store all traffic.



```
#creation d'un fichier journal et on l'ouvre on mode ecriture

fc = open('journal.txt', 'a') #ouverture en mode ajout

fc.write('Journal de capture du : ' + datetime.utcnow().strftime('%d/%m/%y %H:%M:%S.%f')+'\n')

#fc.close() on ne le ferme pas on le laisse ouvert pour y rajouter les autres ligne de capture
```

Figure 4 Python – log of traffic

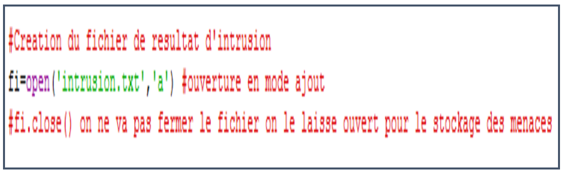
Then we proceed with the loading of our security rules.



```
#On recupere la regle d'intrusion
fr = open('regle.txt','r') #ouverture en mode lecture
regle=fr.readline()
fr.close()
```

Figure 5 Python – loading of the security rules

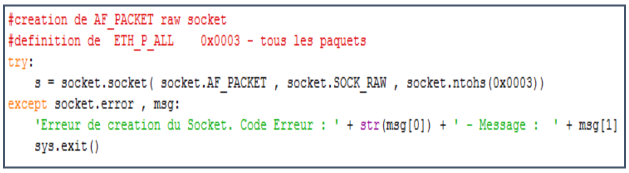
After that comes the step of the creation of the intrusion file.



```
#Creation du fichier de resultat d'intrusion

fi=open('intrusion.txt','a') #ouverture en mode ajout

#fi.close() on ne va pas fermer le fichier on le laisse ouvert pour le stockage des menaces
```

Figure 6 Python – Intrusion file creation

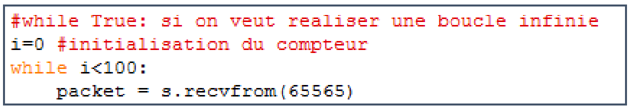
The build of the raw socket is made using the code below.



```
#creation de AF_PACKET raw socket
#definition de  ETH_P_ALL    0x0003 - tous les paquets
try:
    s = socket.socket( socket.AF_PACKET , socket.SOCK_RAW , socket.ntohs(0x0003))
except socket.error , msg:
    'Erreur de creation du Socket. Code Erreur : ' + str(msg[0]) + ' - Message :  ' + msg[1]
    sys.exit()
```

Figure 7 Python –RAW SOCKET build

And capturing traffic starts using the following command:

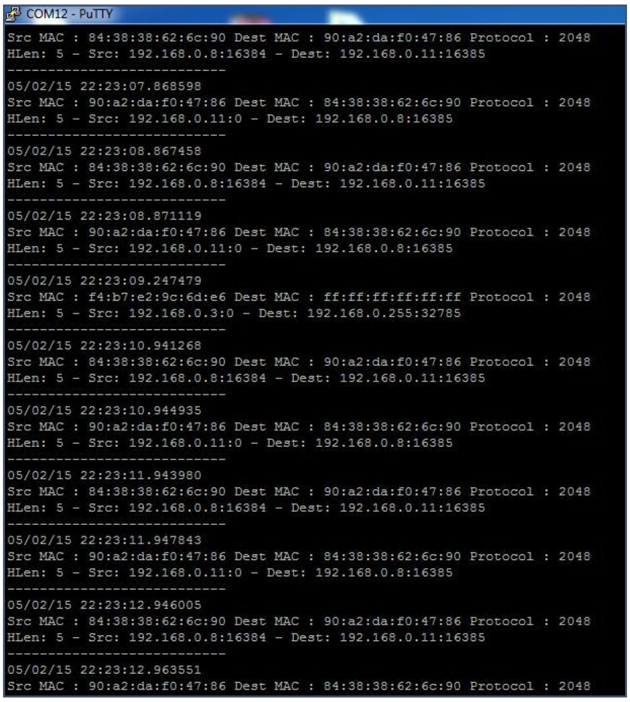


```
#while True: si on veut realiser une boucle infinie
i=0 #initialisation du compteur
while i<100:
    packet = s.recvfrom(65565)
```

Figure 8 Python – Traffic Capturing

We can opt for an infinite loop capture as we can limit the number of captured packets.

The following figure shows how the capture looks like on the PuTTY's software console window.

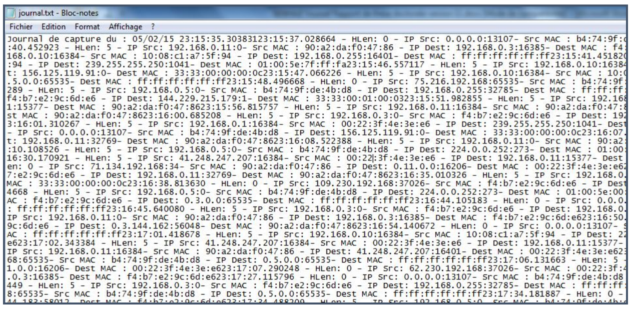


```
Src MAC : 84:38:38:62:6c:90 Dest MAC : 90:a2:da:f0:47:86 Protocol : 2048
HLen: 5 - Src: 192.168.0.8:16384 - Dest: 192.168.0.11:16385
----------------------------
05/02/15 22:23:07.868598
Src MAC : 90:a2:da:f0:47:86 Dest MAC : 84:38:38:62:6c:90 Protocol : 2048
HLen: 5 - Src: 192.168.0.11:0 - Dest: 192.168.0.8:16385
----------------------------
05/02/15 22:23:08.867458
Src MAC : 84:38:38:62:6c:90 Dest MAC : 90:a2:da:f0:47:86 Protocol : 2048
HLen: 5 - Src: 192.168.0.8:16384 - Dest: 192.168.0.11:16385
----------------------------
05/02/15 22:23:08.871119
Src MAC : 90:a2:da:f0:47:86 Dest MAC : 84:38:38:62:6c:90 Protocol : 2048
HLen: 5 - Src: 192.168.0.11:0 - Dest: 192.168.0.8:16385
----------------------------
05/02/15 22:23:09.247479
Src MAC : f4:b7:e2:9c:6d:e6 Dest MAC : ff:ff:ff:ff:ff:ff Protocol : 2048
HLen: 5 - Src: 192.168.0.3:0 - Dest: 192.168.0.255:32785
----------------------------
05/02/15 22:23:10.941268
Src MAC : 84:38:38:62:6c:90 Dest MAC : 90:a2:da:f0:47:86 Protocol : 2048
HLen: 5 - Src: 192.168.0.8:16384 - Dest: 192.168.0.11:16385
----------------------------
05/02/15 22:23:10.944935
Src MAC : 90:a2:da:f0:47:86 Dest MAC : 84:38:38:62:6c:90 Protocol : 2048
HLen: 5 - Src: 192.168.0.11:0 - Dest: 192.168.0.8:16385
----------------------------
05/02/15 22:23:11.943980
Src MAC : 84:38:38:62:6c:90 Dest MAC : 90:a2:da:f0:47:86 Protocol : 2048
HLen: 5 - Src: 192.168.0.8:16384 - Dest: 192.168.0.11:16385
----------------------------
05/02/15 22:23:11.947843
Src MAC : 90:a2:da:f0:47:86 Dest MAC : 84:38:38:62:6c:90 Protocol : 2048
HLen: 5 - Src: 192.168.0.11:0 - Dest: 192.168.0.8:16385
----------------------------
05/02/15 22:23:12.946005
Src MAC : 84:38:38:62:6c:90 Dest MAC : 90:a2:da:f0:47:86 Protocol : 2048
HLen: 5 - Src: 192.168.0.8:16384 - Dest: 192.168.0.11:16385
----------------------------
05/02/15 22:23:12.963551
Src MAC : 90:a2:da:f0:47:86 Dest MAC : 84:38:38:62:6c:90 Protocol : 2048
```

Figure 9 PuTTY – Console Window

We display, time, MAC addresses, IP addresses, ports... Other items can be displayed but we did not want to clutter up the display.

The log file in turn is as follows:

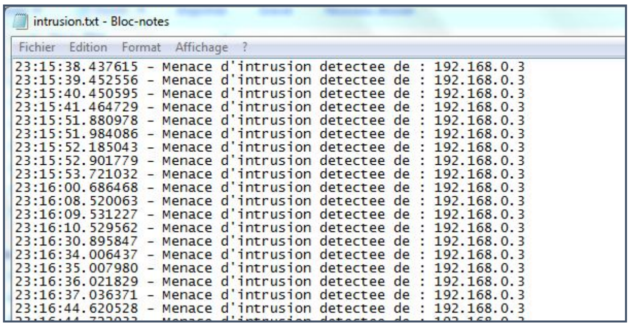


```
Journal de capture du : 05/02/15 23:15:35.30383123:15:37.028664 - HLen: 0 - IP Src: 0.0.0.0:13107- Src MAC : b4:74:9f:d
:40.452923 - HLen: 5 - IP Src: 192.168.0.11:0- Src MAC : 90:a2:da:f0:47:86 - IP Dest: 192.168.0.3:16385- Dest MAC : f4:
168.0.10:16384- Src MAC : 10:08:c1:a7:5f:94 - IP Dest: 192.168.0.255:16401- Dest MAC : ff:ff:ff:ff:ff:ff23:15:41.451820
:94 - IP Dest: 239.255.255.250:1041- Dest MAC : 01:00:5e:7f:ff:fa23:15:46.557117 - HLen: 5 - IP Src: 192.168.0.10:16384
t: 156.125.119.91:0- Dest MAC : 33:33:00:00:00:0c23:15:47.066226 - HLen: 5 - IP Src: 192.168.0.10:16384- Src MAC : 10:0
.5.0.0:65535- Dest MAC : ff:ff:ff:ff:ff:ff23:15:48.496668 - HLen: 0 - IP Src: 75.216.192.168:65535- Src MAC : b4:74:9f:
289 - HLen: 5 - IP Src: 192.168.0.5:0- Src MAC : b4:74:9f:de:4b:d8 - IP Dest: 192.168.0.255:32785- Dest MAC : ff:ff:ff:
f4:b7:e2:9c:6d:e6 - IP Dest: 144.229.215.179:1- Dest MAC : 33:33:00:01:00:0323:15:51.982855 - HLen: 5 - IP Src: 192.168
1:15377- Dest MAC : 90:a2:da:f0:47:8623:15:56.815757 - HLen: 5 - IP Src: 192.168.0.11:16384- Src MAC : 90:a2:da:f0:47:8
st MAC : 90:a2:da:f0:47:8623:16:00.685208 - HLen: 5 - IP Src: 192.168.0.3:0- Src MAC : f4:b7:e2:9c:6d:e6 - IP Dest: 192
3:16:01.310267 - HLen: 5 - IP Src: 192.168.0.1:16384- Src MAC : 00:22:3f:4e:3e:e6 - IP Dest: 239.255.255.250:1041- Dest
- IP Src: 0.0.0.0:13107- Src MAC : b4:74:9f:de:4b:d8 - IP Dest: 156.125.119.91:0- Dest MAC : 33:33:00:00:00:0c23:16:07.
t: 192.168.0.11:32769- Dest MAC : 90:a2:da:f0:47:8623:16:08.522388 - HLen: 5 - IP Src: 192.168.0.11:0- Src MAC : 90:a2:
:10.108526 - HLen: 5 - IP Src: 192.168.0.5:0- Src MAC : b4:74:9f:de:4b:d8 - IP Dest: 224.0.0.252:273- Dest MAC : 01:00:
16:30.170921 - HLen: 5 - IP Src: 41.248.247.207:16384- Src MAC : 00:22:3f:4e:3e:e6 - IP Dest: 192.168.0.11:15377- Dest
en: 0 - IP Src: 71.134.192.168:34- Src MAC : 90:a2:da:f0:47:86 - IP Dest: 0.11.0.0:16206- Dest MAC : 00:22:3f:4e:3e:e62
7:e2:9c:6d:e6 - IP Dest: 192.168.0.11:32769- Dest MAC : 90:a2:da:f0:47:8623:16:35.010326 - HLen: 5 - IP Src: 192.168.0.
MAC : 33:33:00:00:00:0c23:16:38.813630 - HLen: 0 - IP Src: 109.230.192.168:37026- Src MAC : f4:b7:e2:9c:6d:e6 - IP Dest
4668 - HLen: 5 - IP Src: 192.168.0.5:0- Src MAC : b4:74:9f:de:4b:d8 - IP Dest: 224.0.0.252:273- Dest MAC : 01:00:5e:00:
AC : f4:b7:e2:9c:6d:e6 - IP Dest: 0.3.0.0:65535- Dest MAC : ff:ff:ff:ff:ff:ff23:16:44.105183 - HLen: 0 - IP Src: 0.0.0.
: ff:ff:ff:ff:ff:ff23:16:45.640080 - HLen: 5 - IP Src: 192.168.0.3:0- Src MAC : f4:b7:e2:9c:6d:e6 - IP Dest: 192.168.0.
IP Src: 192.168.0.11:0- Src MAC : 90:a2:da:f0:47:86 - IP Dest: 192.168.0.3:16385- Dest MAC : f4:b7:e2:9c:6d:e623:16:50.
9c:6d:e6 - IP Dest: 0.3.144.162:56048- Dest MAC : 90:a2:da:f0:47:8623:16:54.140672 - HLen: 0 - IP Src: 0.0.0.0:13107- S
AC : ff:ff:ff:ff:ff:ff23:17:01.418678 - HLen: 5 - IP Src: 192.168.0.10:16384- Src MAC : 10:08:c1:a7:5f:94 - IP Dest: 22
e623:17:02.343384 - HLen: 5 - IP Src: 41.248.247.207:16384- Src MAC : 00:22:3f:4e:3e:e6 - IP Dest: 192.168.0.11:15377-
IP Src: 192.168.0.11:16384- Src MAC : 90:a2:da:f0:47:86 - IP Dest: 41.248.247.207:16401- Dest MAC : 00:22:3f:4e:3e:e623
68:65535- Src MAC : b4:74:9f:de:4b:d8 - IP Dest: 0.5.0.0:65535- Dest MAC : ff:ff:ff:ff:ff:ff23:17:06.131663 - HLen: 5 -
1.0.0:16206- Dest MAC : 00:22:3f:4e:3e:e623:17:07.290248 - HLen: 0 - IP Src: 62.230.192.168:37026- Src MAC : 00:22:3f:4
.0.3:16385- Dest MAC : f4:b7:e2:9c:6d:e623:17:27.115796 - HLen: 0 - IP Src: 0.0.0.0:13107- Src MAC : b4:74:9f:de:4b:d8
449 - HLen: 5 - IP Src: 192.168.0.3:0- Src MAC : f4:b7:e2:9c:6d:e6 - IP Dest: 192.168.0.255:32785- Dest MAC : ff:ff:ff:
8:65535- Src MAC : b4:74:9f:de:4b:d8 - IP Dest: 0.5.0.0:65535- Dest MAC : ff:ff:ff:ff:ff:ff23:17:34.181887 - HLen: 0 -
44.183:58012- Dest MAC : f4:b7:e2:9c:6d:e623:17:34.488209 - HLen: 5 - IP Src: 192.168.0.5:0- Src MAC : b4:74:9f:de:4b:d
```

Figure 10 log file

This is the same data displayed in the PuTTY's console Window but without carriage returns.

Our security rule is an IP address of intrusion. As soon as our scanner detects a threat, it saves it in the intrusion file.



intrusion.txt - Bloc-notes
Fichier Edition Format Affichage ?

```
23:15:38.437615 - Menace d'intrusion detectee de : 192.168.0.3
23:15:39.452556 - Menace d'intrusion detectee de : 192.168.0.3
23:15:40.450595 - Menace d'intrusion detectee de : 192.168.0.3
23:15:41.464729 - Menace d'intrusion detectee de : 192.168.0.3
23:15:51.880978 - Menace d'intrusion detectee de : 192.168.0.3
23:15:51.984086 - Menace d'intrusion detectee de : 192.168.0.3
23:15:52.185043 - Menace d'intrusion detectee de : 192.168.0.3
23:15:52.901779 - Menace d'intrusion detectee de : 192.168.0.3
23:15:53.721032 - Menace d'intrusion detectee de : 192.168.0.3
23:16:00.686468 - Menace d'intrusion detectee de : 192.168.0.3
23:16:08.520063 - Menace d'intrusion detectee de : 192.168.0.3
23:16:09.531227 - Menace d'intrusion detectee de : 192.168.0.3
23:16:10.529562 - Menace d'intrusion detectee de : 192.168.0.3
23:16:30.895847 - Menace d'intrusion detectee de : 192.168.0.3
23:16:34.006437 - Menace d'intrusion detectee de : 192.168.0.3
23:16:35.007980 - Menace d'intrusion detectee de : 192.168.0.3
23:16:36.021829 - Menace d'intrusion detectee de : 192.168.0.3
23:16:37.036371 - Menace d'intrusion detectee de : 192.168.0.3
23:16:44.620528 - Menace d'intrusion detectee de : 192.168.0.3
23:16:44.723033 - Menace d'intrusion detectee de : 192.168.0.3
```

Figure 11 Intrusion file

We can automate the launch of our analysis program with the boot of our board by changing the LININO boot file. The latter has the path:

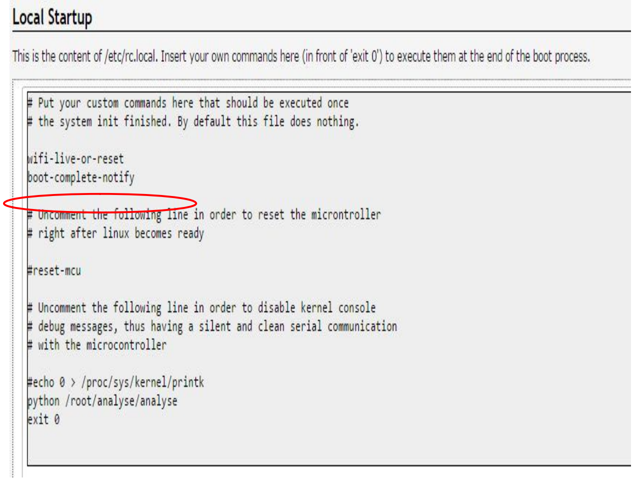
/etc/rc.local

The modification consists of launching our "analysis" program using the command:

python /root/analyse/analyse

This can be done in console mode or through the GUI LUCI of the Arduino board.

The figure below shows this approach with LUCI:



Local Startup

This is the content of /etc/rc.local. Insert your own commands here (in front of 'exit 0') to execute them at the end of the boot process.

```
# Put your custom commands here that should be executed once
# the system init finished. By default this file does nothing.

wifi-live-or-reset
boot-complete-notify

# Uncomment the following line in order to reset the microntroller
# right after linux becomes ready

#reset-mcu

# Uncomment the following line in order to disable kernel console
# debug messages, thus having a silent and clean serial communication
# with the microcontroller

#echo 0 > /proc/sys/kernel/printk
python /root/analyse/analyse
exit 0
```

Figure 12 Automation of the "analysis"

5.2Evaluation

We carry out a series of attacks on our detection system to assess its response time to an attack. Thus we get the results below.

Table 1 Results

Attack Number	attack Instant	attack detection	detection time (ms)
1	14:16:04,043	14:16:04,649	0:00:00,606
2	14:16:12,919	14:16:13,600	0:00:00,681
3	14:16:23,403	14:16:24,027	0:00:00,624
4	14:16:36,756	14:16:37,364	0:00:00,608
5	14:16:52,091	14:16:52,722	0:00:00,631
6	14:17:02,716	14:17:03,355	0:00:00,639
7	14:17:13,792	14:17:14,431	0:00:00,639
8	14:17:25,804	14:17:26,428	0:00:00,624
9	14:17:35,257	14:17:35,882	0:00:00,625
10	14:17:48,736	14:17:49,360	0:00:00,624
11	14:18:02,838	14:18:03,433	0:00:00,595
12	14:18:11,793	14:18:12,418	0:00:00,625
13	14:18:21,948	14:18:22,661	0:00:00,713
14	14:18:37,000	14:18:38,041	0:00:01,041
15	14:19:03,101	14:19:03,711	0:00:00,610
		Average	0:00:00,659
		Min	0:00:00,595
		Max	0:00:01,041

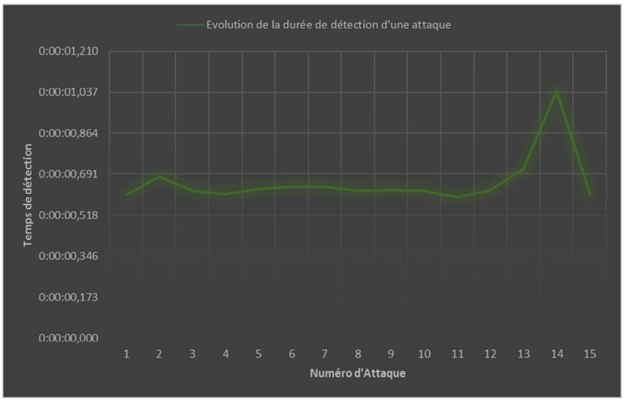


Figure 13 Results

Of course, this detection time may vary depending on:

- The physical characteristics of our simulation system workstations, network cards, Switch ...
- Network saturation at the time of the attack
- The number of attacks
- The duration between attacks
- The number and nature of security rules
- etc.

6.CONCLUSION AND FURTHER WORK

In this paper, we proposed a security architecture based on a distributed approach. The latter combines three Arduino Yun Boards in a semantic distribution based on detection method.

The aim is to reduce the analysis loading to improve response time, to reduce the number of false positives and false negatives, to ensure interoperability between the detection system and the prevention one, to reduce the number of harmless blocking traffic and to clean traces of detection and prevention.

As further work we can detail more the securing of IDPS and the securing of their LOG files by developing a network management controlled by its own firewall. We can also study an additional distribution by protocol “structural distribution” (level 4: Web streaming, FTP streaming, SQL query ...). Moreover, we can think through optimizing safety rules (ex: if the network is running on Windows systems, the rules for UNIX systems are not needed). And at last, we can develop in detail the interface of illustration. We can also study the possibility to create with Arduino Boards a Proxy system to improve the prevention system.

References

[1] Open Information Security Foundation. « Getting Started With Suricata ». OISF, 2011

[2] Karen Scarfone, Peter Mell. “Guide to Intrusion Detection and Prevention Systems IDPS”. NIST. US Departement of Commerce. 2007

[3] Boriana Ditcheva, Lisa Fowler. “Signature-based Intrusion Detection”. University of North Carolina at Chapel Hill. 2005

[4] Rebecca Bace, Peter Mell. "NIST Special Publication on Intrusion Detection Systems". Infidel, Inc., Scotts Valley, CA - National Institute of Standards and Technology. 2003
[5] Rachid Guerraoui, Lu´ıs Rodrigues, "Introduction to reliable distributed programming'', Springer-Verlag, August 24, 2005.
[6] David Burgermeister, Jonathan Krier, "les systèmes de détection d'intrusions'', Developpez.com, July 22, 2006.
[7] Web Site: Arduino - http://www.arduino.cc/.

AUTHORS

Youssef SENHAJI received his Degree in engineering from the ENSAM, Meknès, Morocco. In 2009, he joined the System Architecture Team of the ENSEM School, Casablanca, Morocco. His current main research interests IDPS in a Distributed Multi-agents Systems.

Hicham MEDROMI received his PhD in engineering science from the Sophia Antipolis University in 1996, Nice, France. He is responsible of the system architecture team of the ENSEM Hassan II University, Casablanca, Morocco. His current main research interests concern Control Architecture of Mobile Systems Based on Multi Agents Systems. Since 2003 he is a full professor for automatic productic and computer sciences at the ENSEM School, Hassan II University, Casablanca