

EXAMEN 2

Integrantes:

- Monroy Martos Elioth
- Hernández Pineda Miguel Angel
- Zúñiga Hernández Carlos

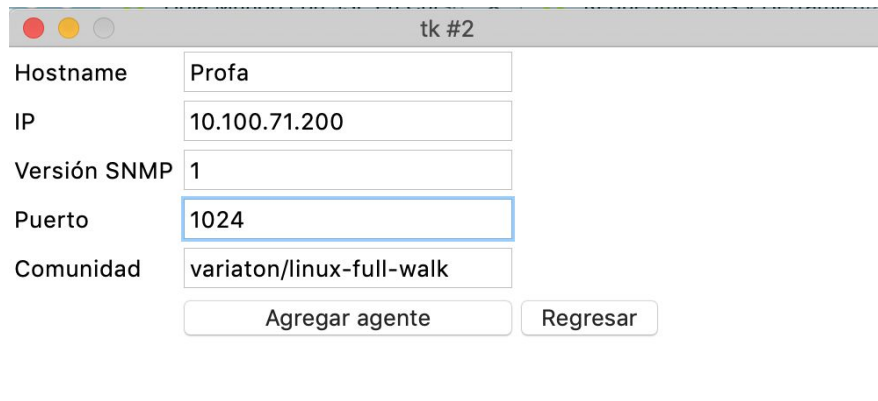
Equipo: 5

Grupo: 4CM1

Parte 1

Evidencia 1

Primero agregamos el agente especificado por la profesora al sistema ingresando su información al sistema para poder realizar un monitoreo sobre él.



Hostname	Profa
IP	10.100.71.200
Versión SNMP	1
Puerto	1024
Comunidad	variaton/linux-full-walk

Posteriormente usando el script (detector_umbrales.py incluido en el zip), que permite realizar mediciones al objeto de la MIB especificado durante un determinado lapso de tiempo y escribir las mismas a un archivo. Para esto especificamos el tiempo que se realizará la medición, el id del agente que agregamos anteriormente y el nombre del OID a monitorear, en este caso hrProcessorLoad.

Las líneas más importantes de este script son las siguientes:

Línea donde se especifica objeto OID y el CPU a monitorear.

```
self.graficarCPU("HOST-RESOURCES-MIB", "hrProcessorLoad", "cpu_umbral", "Uso del  
Procesador", 1281)
```

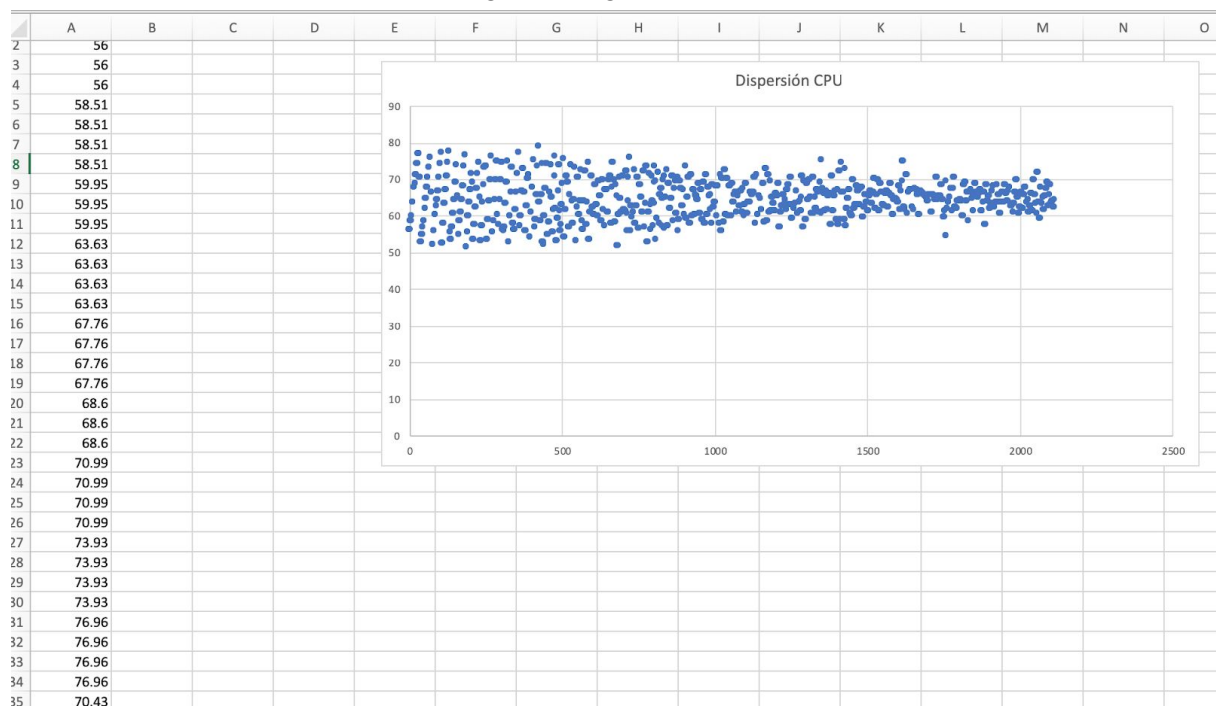
Línea donde se especifica el id del agente y el tiempo de monitoreo, el "4" hace referencia al id asignado al agente al momento de registrarlo, y el tiempo dado en minutos (en este caso 10)

```
detector=DetectorUmbrales("4", 10, 0)  
detector.obtenerUmbrales()
```

Los valores medidos, son escritos en un archivo llamado "umbral_CPU.txt", como se muestra en la siguiente figura:

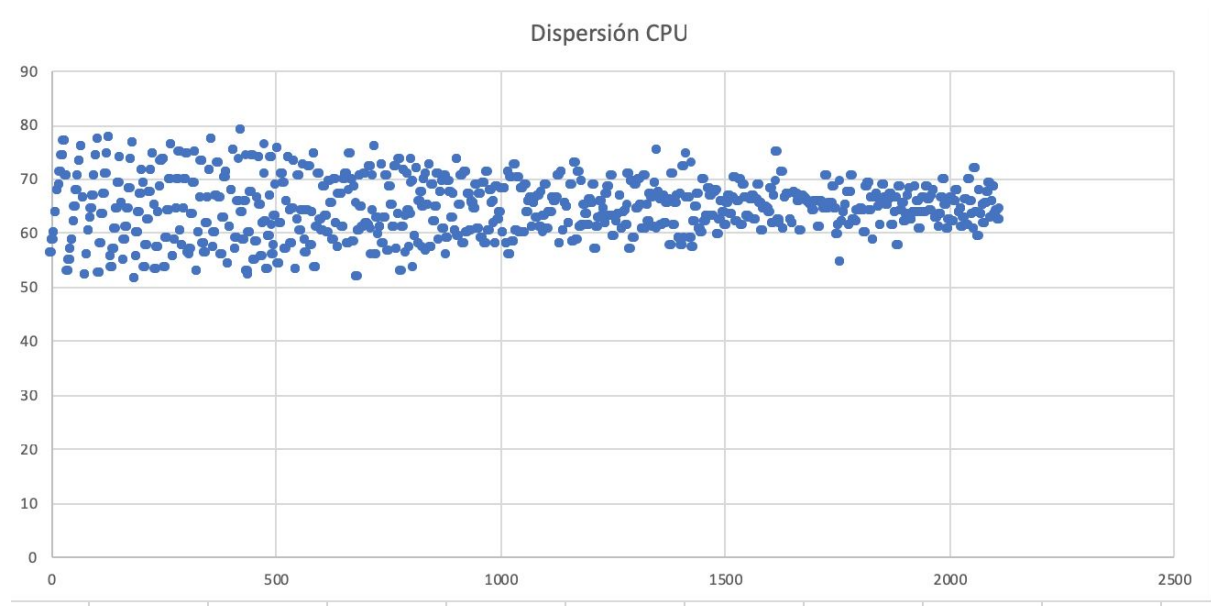
1	56.0
2	56.0
3	56.0
4	56.0
5	58.51
6	58.51
7	58.51
8	58.51
9	59.95
10	59.95
11	59.95
12	63.63
13	63.63
14	63.63
15	63.63
16	67.76
17	67.76
18	67.76
19	67.76
20	68.6
21	68.6
22	68.6
23	70.99
24	70.99
25	70.99
26	70.99
27	73.93
28	73.93
29	73.93
30	73.93

Posteriormente, esos valores, fueron trasladados a excel, para su posterior análisis usando dispersión, como se muestra en la siguiente figura:



Del gráfico de dispersión (siguiente figura), podemos observar que de las más de 2000 muestras tomadas, ninguna logró pasar el valor 80 (que se refiere a 80 por ciento de uso de CPU).

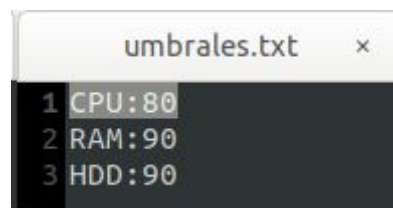
Siendo el máximo valor obtenido 78, por lo cual se determino el umbral GO como **80**.



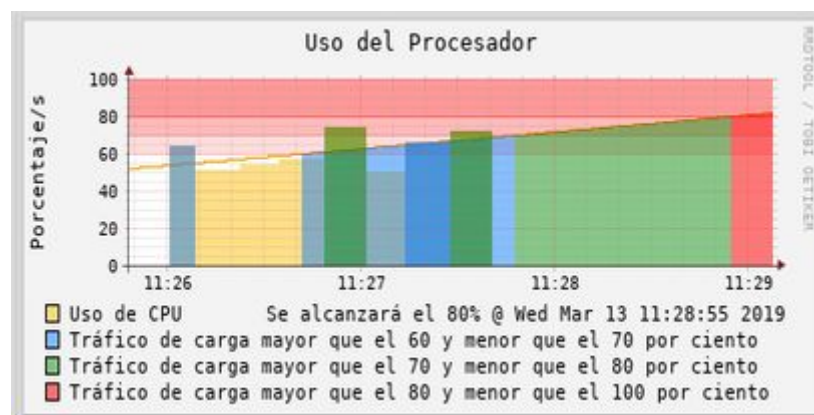
PARTE 2

Evidencia 2

El umbral definido, se indica en el archivo ("umbrales.txt"), ya que en automatico el sistema calcula los siguientes umbrales (Set y Ready), siendo cada uno el umbral GO -20 y GO-10 respectivamente.



El resultado de estos umbrales generados (para este caso GO:80, Ready:70, Set:60) se muestra en la siguiente imagen.



En la gráfica se puede observar que el umbral GO es del 80%. También, en las descripciones de la gráfica se encuentra el tiempo en el que el uso del cpu va a alcanzar ese umbral. Este tiempo es: 13 de marzo del 2019 a las 11:28:55.

La línea de mejor ajuste se muestra de color amarillo. Se puede ver que ésta empieza en el porcentaje aproximado de 50 y va recorriendo la gráfica hasta alcanzar el porcentaje 80, el cual corresponde al umbral GO o máximo.

En cuanto a los colores que se pueden apreciar, el amarillo corresponde al uso del CPU antes de sobrepasar cualquier umbral. El azul corresponde al uso de CPU que sobrepasó el primer umbral que va de 60 a antes del 70 por ciento. El verde corresponde al uso de CPU que sobrepasó el segundo umbral que va del 70 a antes del 80 por ciento y, finalmente, el rojo indica en la gráfica que se va a alcanzar el umbral GO que corresponde del porcentaje 80 al 100.

Para hacer las lecturas de CPU utilizamos el siguiente código, para que en la gráfica se observaran los valores indicados.

```
def graficarCPU(self, grupo, oid, archivo, header, numero):
    umbral=int(self.umbralCPU)
    ultimo=rrdtool.last(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".rrd")
    primero=rrdtool.first(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".rrd")
    consulta=consultav2SNMP(self.agente.comunidad,self.agente.ip,int(self.agente.version),numero,grupo,oid,int(self.agente.puerto))
    if consulta!="":
        valor = "N:" + str(consulta)
        rrdtool.update(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".rrd", valor)
        rrdtool.dump(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".rrd",self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".xml")
        ret = rrdtool.graphv(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".png",
            "--start",str(ultimo-100),
            "--end",str(ultimo+100),
            #"--end",str(ultimo+200),
            "--vertical-label=Porcentaje/s",
            "--title="+header,
            "--lower-limit","0",
            "--upper-limit","100",
            "DEF:usage="+self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".rrd:valor1:AVERAGE",
            "CDEF:umbral"+str(umbral-20)+"=usage,"+str(umbral-20)+",GT,usage,"+str(umbral-10)+",LT,EQ,usage,0,IF",
            "CDEF:umbral"+str(umbral-10)+"=usage,"+str(umbral-10)+",GT,usage,"+str(umbral)+",LT,EQ,usage,0,IF",
            "CDEF:umbral"+str(umbral)+"=usage,"+str(umbral)+",GT,usage,"+str(umbral+(100-umbral))+",LT,EQ,usage,0,IF",
            "AREA:usage#FFB000:Uso de CPU",
            "VDEF:m=usage,LSLSLOPE",
            "VDEF:b=usage,LSLINT",
            "CDEF:avg=usage,POP,m,COUNT,*b,+",
            "CDEF:umbral"+str(umbral-20)+"=avg,"+str(umbral-20)+",LIMIT",
            "CDEF:umbral"+str(umbral-10)+"=avg,"+str(umbral-10)+",LIMIT",
            "CDEF:umbral"+str(umbral)+"=avg,"+str(umbral)+",LIMIT",
            "VDEF:minUmbral"+str(umbral)+"=umbral"+str(umbral)+",FIRST",
            "VDEF:last=usage,LAST",
            "PRINT:last:%6.2lf %s",
            "GPRINT:minUmbral"+str(umbral)+" : Se alcanzará el "+str(umbral)+"% @ %c :strftime",
            "LINE1:avg#FF9900",
            "LINE2:"+str(umbral-20),
            "AREA:10#FF000022::STACK",
            "AREA:10#FF000044::STACK",
            "AREA:"+str(100-umbral)+"#FF000066::STACK",
            "AREA:umbral"+str(umbral-20)+"#007FFF77:Tráfico de carga mayor que el "+str(umbral-20)+" y menor que el "+str(umbral-10)+" por ciento",
            "AREA:umbral"+str(umbral-10)+"#00808077:Tráfico de carga mayor que el "+str(umbral-10)+" y menor que el "+str(umbral)+" por ciento",
            "AREA:umbral"+str(umbral)+"#FF008088:Tráfico de carga mayor que el "+str(umbral)+" y menor que el "+str(umbral+(100-umbral))+ por ciento",
            "AREA:umbral"+str(umbral-20)+"_1#007FFF77",
            "AREA:umbral"+str(umbral-10)+"_1#00808077",
            "AREA:umbral"+str(umbral)+"_1#FF008088")
        valor=float(ret["print[0]"])
        if valor>float(umbral):
            if not self.notificacionCPU:
                self.notificacionCPU=True
                self.noti.enviarCorreo(0,self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".png")
                self.log.escribirLog(0)
            else:
                self.notificacionCPU=False
```

PARTE 3

Evidencia 3

En esta parte se envió un correo a la dirección "tanibet.escom@gmail.com", el correo enviado se puede observar en la siguiente imagen.

Sobrecarga de CPU detectada, Evidencia 3 - Equipo 5 - 4CM1



admiredes3.4cm1@gmail.com

11:38 (hace 2 minutos)



El agente: profa , ha sobrepasado el umbral máximo de uso de CPU, a las: 2019-03-13 11:38:12.943850



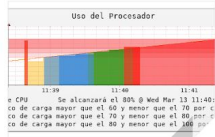
admiredes3.4cm1@gmail.com

11:40 (hace 0 minutos)



para tanibet.escom

El agente: profa , ha sobrepasado el umbral máximo de uso de CPU, a las:
2019-03-13 11:40:18.820793



Responder

Reenviar

El correo fue enviado en tiempo real, al momento que se detectó que una de las mediciones superó el umbral GO anteriormente definido en **80**.

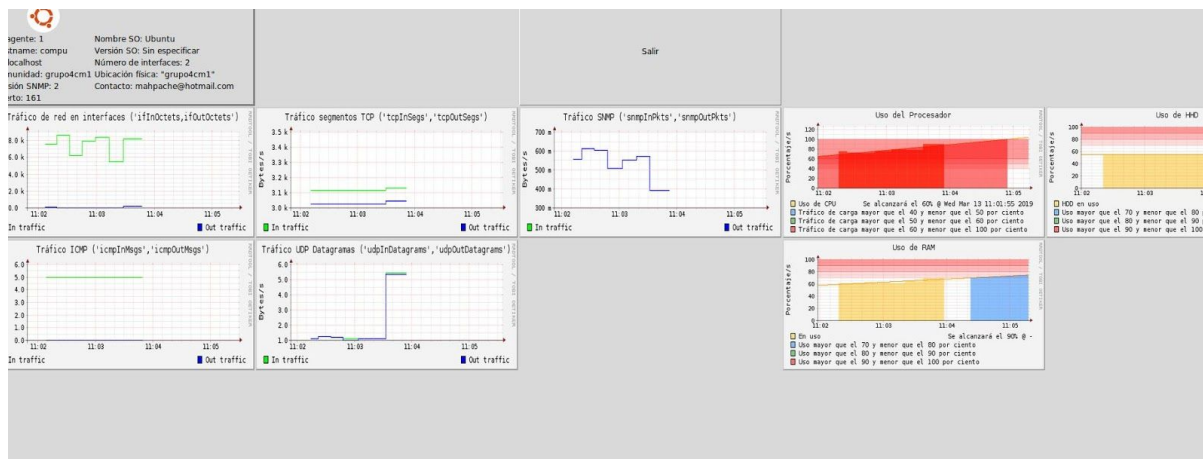
Para enviar el correo se usó notificador.py, como se puede observar en la siguiente imagen:

```
class Notificador():
    def __init__(self, agente):
        self.remitente = "admiredes3.4cm1@gmail.com"
        self.destinatario = "tanibet.escom@gmail.com"
        self.servidor = 'smtp.gmail.com: 587'
        self.contra = '#Equipo5'
        self.agente=agente
    def enviarCorreo(self, tipo, imagen=""):
        asunto, texto=self.obtenerContenido(tipo)
        hora=self.obtenerHora()
        texto="El agente: "+self.agente+" "+texto+hora
        #Definimos contenido del correo
        mensaje = MIMEText(texto, "plain")
        mensaje['Subject'] = asunto
        mensaje['From'] = self.remitente
        mensaje['To'] = self.destinatario
        mensaje.attach(MIMEText(texto, "plain"))
        #Verificamos si hay que enviar una imagen
        if imagen!="":
            with open(imagen, 'rb') as f:
                img = MIMEImage(f.read())
                f.close()
                mensaje.attach(img)
        #Enviamos el correo
        servidor_correo = smtplib.SMTP(self.servidor)
        servidor_correo.starttls()
        servidor_correo.login(self.remitente, self.contra)
        servidor_correo.sendmail(self.remitente, self.destinatario, mensaje.as_string())
        servidor_correo.quit()
    def obtenerContenido(self, tipo):
        with open("mensajes.txt") as f:
            for i, linea in enumerate(f):
                if i==tipo:
                    return linea.split(";")
                else:
                    return ["", ""]
    def obtenerHora(self):
        return str(datetime.datetime.now())
```

Para poder enviar la notificación al correo, se utilizó el código que se puede observar en la parte anterior.

Evidencia 4

Para esta parte se analizó un agente Linux, del cual obtuvimos los resultados del procesamiento de RAM, HDD y CPU. Los datos del agente son: sistema operativo Ubuntu, con comunidad grupo4cm1 y con ip 127.0.0.1 o localhost.



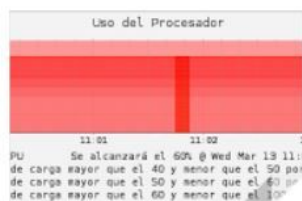
Aquí se envió la notificación para CPU y como podemos en RAM se hace la predicción de llegada al umbral de 60% de uso.

Sobrecarga de CPU detectada Recibidos x



admiredes3.4cm1@gmail.com
para admiredes3.4CM1 ▼

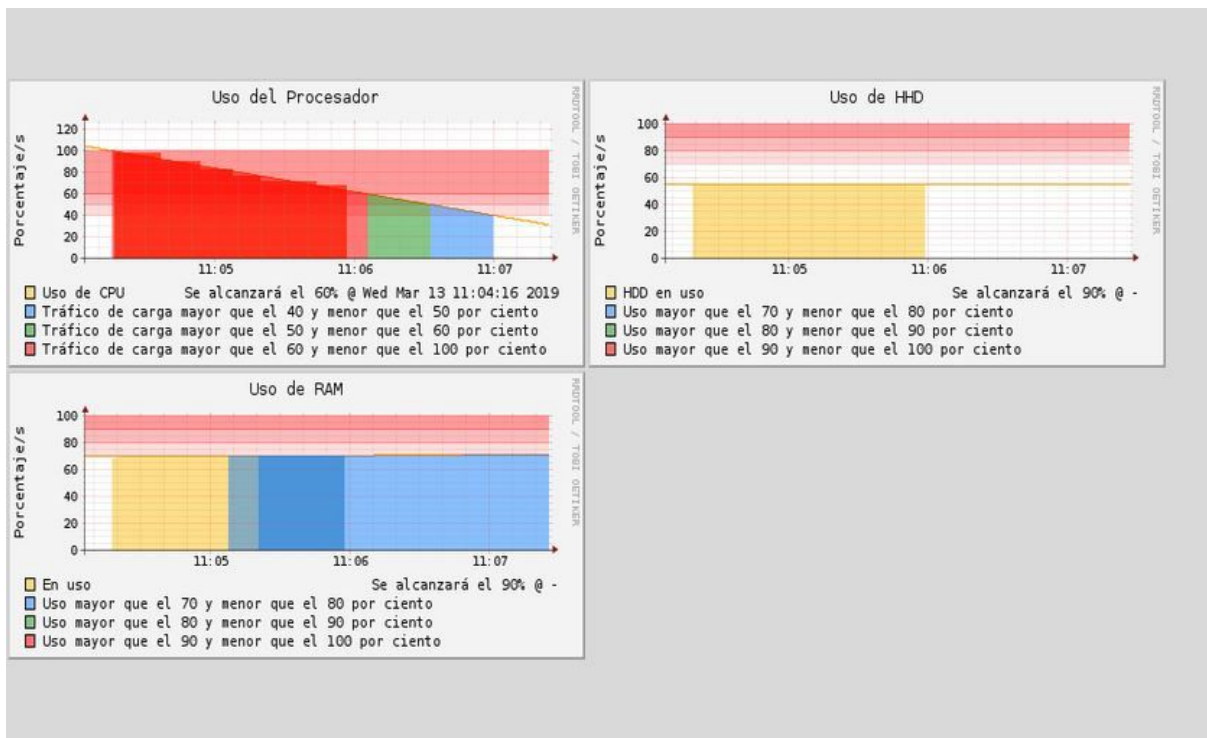
El agente: compu , ha sobrepasado el umbral máximo de uso de CPU, a las:
2019-03-13 11:01:53.728009



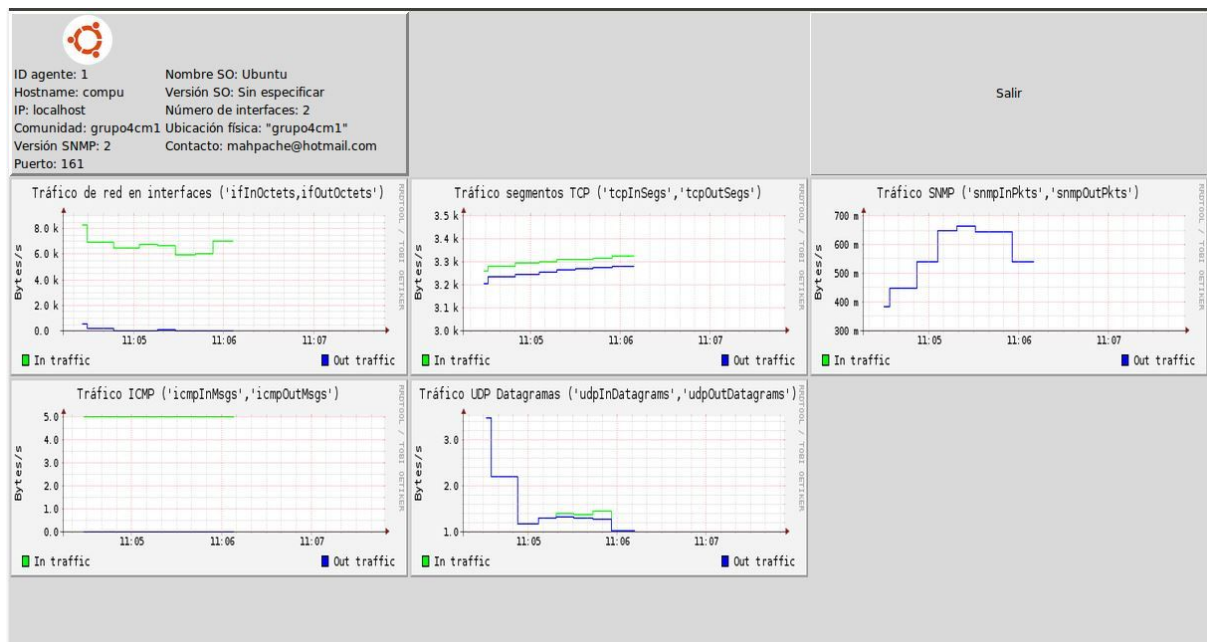
← Responder

➡ Reenviar

Correo recibido como notificación del paso de umbral para CPU.



Después de un tiempo el uso de la RAM llegó a 60 % como se puede ver en la imagen, y el procesador empezó a disminuir como muestra la gráfica.



Además, se obtuvieron las gráficas para el tráfico de interfaces, segmentos TCP, tráfico SNMP, tráfico ICMP, y tráfico de datagramas UDP.