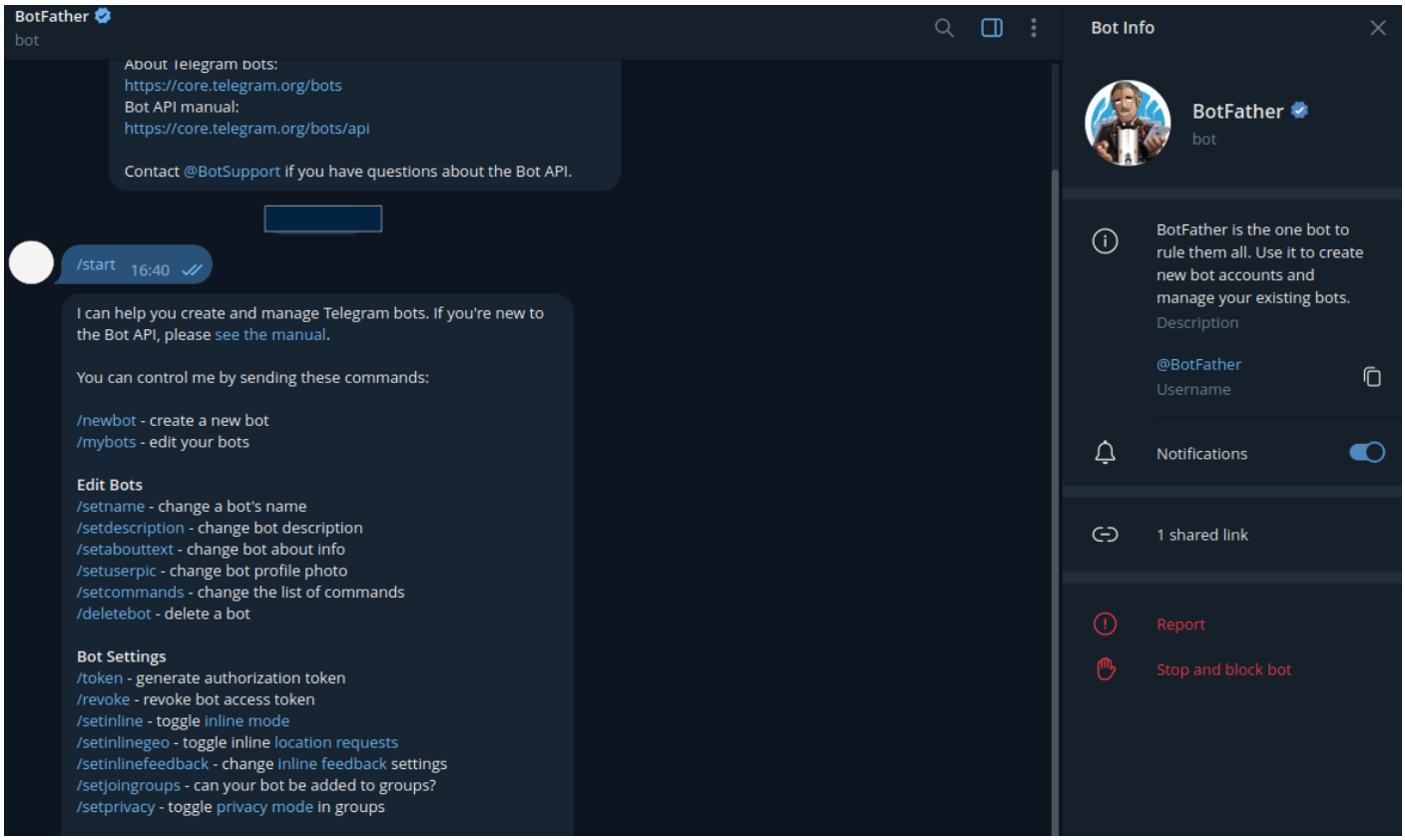
Envío de alertas mediante bot de Telegram

En esta publicación, crearemos un bot de Telegram para recibir alertas de Wazuh directamente en un chat de Telegram usando el módulo de integraciones en Wazuh.

Como primer paso crearemos un bot de Telegram con la ayuda de BotFather.



Para crear el bot seguimos los siguientes comandos:
Crear bot:
`/newbot`

Nos pedira darle un nombre al bot:
`wazuhbot2050_bot`

Nos dara un número nuevo de API para el bot, la respuesta que en este caso es:
`ClaveAPI2050Alfanumerica`

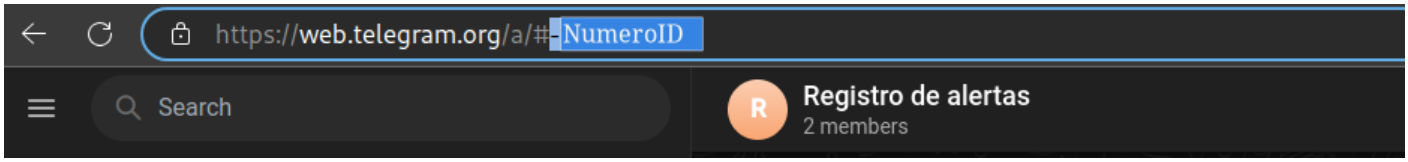
La clave API es muy importante por que se incluirá a los scripts de Wazuh manager y se utiliza para obtener el chat ID.

Utilizaremos un script en Wazuh manager de la siguiente manera:

```
sudo nano /var/ossec/integrations/custom-telegram
```

```
#!/bin/sh
WPYTHON_BIN="framework/python/bin/python3"
SCRIPT_PATH_NAME="$0"
DIR_NAME="$(cd $(dirname ${SCRIPT_PATH_NAME}); pwd -P)"
SCRIPT_NAME="$(basename ${SCRIPT_PATH_NAME})"
case ${DIR_NAME} in
  */active-response/bin | */wodles*)
    if [ -z "${WAZUH_PATH}" ]; then
      WAZUH_PATH="$(cd ${DIR_NAME}/../..; pwd)"
    fi
    PYTHON_SCRIPT="${DIR_NAME}/${SCRIPT_NAME}.py"
  ;;
  */bin)
    if [ -z "${WAZUH_PATH}" ]; then
      WAZUH_PATH="$(cd ${DIR_NAME}/..; pwd)"
    fi
    PYTHON_SCRIPT="${WAZUH_PATH}/framework/scripts/${SCRIPT_NAME}.py"
  ;;
   */integrations)
    if [ -z "${WAZUH_PATH}" ]; then
      WAZUH_PATH="$(cd ${DIR_NAME}/..; pwd)"
    fi
    PYTHON_SCRIPT="${DIR_NAME}/${SCRIPT_NAME}.py"
  ;;
esac
${WAZUH_PATH}/${WPYTHON_BIN} ${PYTHON_SCRIPT} "$@"
```

Si deseamos que el registro de alertas se envíe a un grupo en específico debemos ingresar a telegram web desde el navegador y copiar el ID de grupo al script de python incluyendo el signo “-”.



Accediendo desde Telegram web se puede observar el numero ID de grupo que se genera, dato que necesitaremos posteriormente para reenviar las alertas al grupo ya que es mas conveniente en caso de tener un incidente de gran escala que varias personas se enteren, En el inicialmente se encuentra el bot y el numero de la persona que realiza la integración.

Configuramos el script de telegram con el dato extra del grupo para realizar la función del script.

```
#!/usr/bin/env python
import sys
import json
import requests
from requests.auth import HTTPBasicAuth
#CHAT_ID="xxxx"
CHAT_ID="-NumeroID"
# Read configuration parameters
alert_file = open(sys.argv[1])
hook_url = sys.argv[3]

# Read the alert file
alert_json = json.loads(alert_file.read())
alert_file.close()
# Extract data fields
alert_level = alert_json['rule']['level'] if 'level' in alert_json['rule'] else "N/A"
description = alert_json['rule']['description'] if 'description' in alert_json['rule'] else "N/A"
agent = alert_json['agent']['name'] if 'name' in alert_json['agent'] else "N/A"
username = alert_json['data']['dstuser'] if 'dstuser' in alert_json['data'] else "N/A"
source = alert_json['data']['scrip'] if 'scrip' in alert_json['data'] else "N/A"
# Generate request
msg_data = {}
msg_data['chat_id'] = CHAT_ID
msg_data['text'] = {}
msg_data['text']['description'] =  description
msg_data['text']['alert_level'] = str(alert_level)
msg_data['text']['agent'] =  agent
```

```
msg_data['text']['username'] =  username
msg_data['text']['source'] =  source
headers = {'content-type': 'application/json', 'Accept-Charset': 'UTF-8'}
# Send the request
requests.post(hook_url, headers=headers, data=json.dumps(msg_data))
sys.exit(0)
```

```
nano /var/ossec/integrations/custom-telegram.py
```

En los scripts iniciales el grupo que administraba las operaciones de las integraciones con wazuh era “ossec” sin embargo desde la version 4.4.5 el grupo que administra estos eventos es “wazuh”

```
chown root:wazuh /var/ossec/integrations/custom-telegram*
chmod 750 /var/ossec/integrations/custom-telegram*
```

Es importante mencionar que en este caso se trata de una integración externa personalizada ya que Telegram no se encuentra entre las pruebas de concepto y el nombre de referenciación debe comenzar con “custom-”.
Las configuraciones se realizan en el archivo ossec.conf. Se pueden utilizar los siguientes parámetros:

- **name**: Nombre del script que realiza la integración. En el caso de una integración personalizada como la que se analiza en este artículo, el nombre debe comenzar con "personalizado-".
- **hook_url**: URL proporcionada por la API del software para conectarse a la propia API. Su uso es opcional, ya que se puede incluir en el script.
- **api_key** : Clave de la API que nos permite utilizarla. Su uso también es opcional por la misma razón que el uso de hook_url es opcional.
- **level** : establece un filtro de nivel para que el script no reciba alertas por debajo de cierto nivel.
- **rule_id** : establece un filtro para los identificadores de alerta.
- **group** : Establece un filtro de grupo de alertas.
- **event_location** : establece un filtro de fuente de alerta.
- **alert_format** : Indica que el script recibe las alertas en formato JSON (recomendado). De forma predeterminada, el script recibirá las alertas en formato full_log.

La configuración que realizaremos en Wazuh manager se da de la siguiente manera:

```
nano /var/ossec/etc/ossec.conf
```

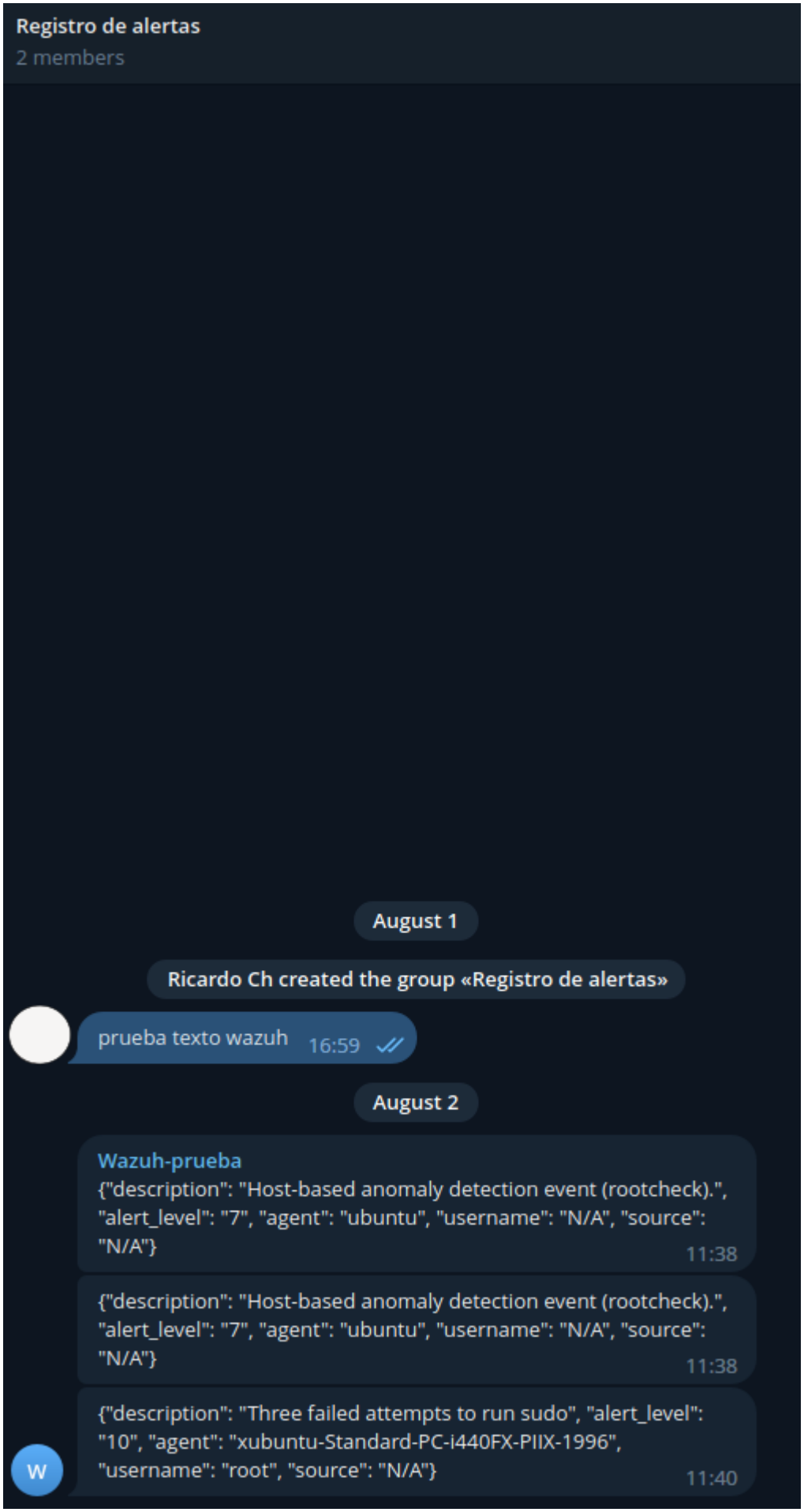
```
  <integration>
    <name>custom-telegram</nasystemctl restart wazuh-managerme>
    <level>7</level> <hook_url>https://api.telegram.org/botClaveAPI2050Alfanumerica/sendMessage</hook_url>
    <alert_format>json</alert_format>
  </integration>
```

Para que los cambios se apliquen realizamos un restart.

```
systemctl restart wazuh-manager
```

Es recomendable asignar el rol de administrador al bot creado, ya que sin realizar esta acción no podrá enviar mensajes, pues está restringido por defecto.

Una vez realizadas las configuraciones, el grupo creado en Telegram con el bot será el que reciba las alertas y podrá ser visualizado desde un rango establecido por el nivel en las vulnerabilidades como se muestra a continuación.


Registro de alertas
2 members
August 1
Ricardo Ch created the group «Registro de alertas»
prueba texto wazuh 16:59
August 2
Wazuh-prueba
{"description": "Host-based anomaly detection event (rootcheck).", "alert_level": "7", "agent": "ubuntu", "username": "N/A", "source": "N/A"}
11:38
{"description": "Host-based anomaly detection event (rootcheck).", "alert_level": "7", "agent": "ubuntu", "username": "N/A", "source": "N/A"}
11:38
{"description": "Three failed attempts to run sudo", "alert_level": "10", "agent": "xubuntu-Standard-PC-i440FX-PIIX-1996", "username": "root", "source": "N/A"}
11:40
W

Revision #5
Created 9 abril 2024 10:53:06 by Ricardo Chavez
Updated 10 abril 2024 12:16:47 by Ricardo Chavez