Práctica de laboratorio: Convertir datos a un formato universal Objetivos

Parte 1: Normalizar marcas de hora en un archivo de registro

Parte 2: Normalizar marcas de hora en un archivo de registro Apache

Parte 3: Preparar de archivos de registro en Security Onion

Aspectos básicos/Situación

Esta práctica de laboratorio preparará a los alumnos para aprender dónde se encuentran los archivos de registro, cómo manipularlos y verlos. Las **entradas de registro** son generadas por dispositivos de red, sistemas operativos, aplicaciones y diversos tipos de dispositivos programables. A un archivo que contiene una secuencia temporizada de entradas de registro se le llama **archivo de registro**.

Por naturaleza, los archivos de registro registran eventos que son relevantes a la fuente. La sintaxis y el formato de los datos que se encuentran dentro de los mensajes de registro a menudo son definidos por el desarrollador de la aplicación.

Por lo tanto, la terminología utilizada en las entradas de registro a menudo varía según la fuente. Por ejemplo: dependiendo de la fuente, los términos "iniciar sesión", "conectarse", "evento de autenticación" y "conexión del usuario" pueden aparecer en entradas de registro para describir la autenticación exitosa de un usuario en un servidor.

A menudo es aconsejable tener terminología consistente y uniforme en los archivos de registro generados por diferentes fuentes. Esto es especialmente cierto cuando todos los archivos de registro son recopilados por un punto centralizado.

El término **normalización** se refiere al proceso de convertir partes de un mensaje, en este caso una entrada de registro, a un formato común.

En esta práctica de laboratorio utilizarán herramientas de la línea de comando para normalizar las entradas de registro manualmente. En la Parte 2, se normalizará el campo de la marca de tiempo. En la Parte 3, se normalizará el campo de IPv6.

Nota: Si bien hay numerosos complementos para realizar la normalización de archivos de registro, es importante comprender los fundamentos básicos que subyacen al proceso de normalización.

Recursos necesarios

- Máquina virtual CyberOps Workstation
- Máquina virtual de Security Onion

Instrucciones

Parte 1: Normalizar marcas de hora en un archivo de registro

Las marcas de hora se utilizan en entradas de registro para especificar cuándo tuvo lugar el evento registrado. Si bien la práctica recomendada es registrar marcas de hora en UTC, el formato de las marcas de hora varía según la fuente del archivo de registro. Hay dos formatos de marca de hora comunes; se conocen como Unix Epoch y Human Readable (Legible por seres humanos).

Las marcas de tiempo Unix Epoch registran la hora midiendo la cantidad de segundos transcurridos desde el 1 de enero de 1970.

Las marcas de hora legibles registran la hora representando valores aparte para el año, el mes, el día, las horas, los minutos y los segundos.

La marca de hora legible **Wed, 28 Jun 2017 13:27:18 GMT** equivale a **1498656439** en Unix Epoch.

Desde un punto de vista de programabilidad, es mucho más fácil trabajar con Epoch, ya que permite facilitar operaciones de suma y resta. Desde una perspectiva de análisis; sin embargo, las marcas de hora legibles son mucho más fáciles de interpretar.

Convertir marcas de hora Epoch a marcas de hora legibles con AWK

AWK es un lenguaje de programación diseñado para manipular archivos de texto. Es muy potente y especialmente útil para manejar archivos de texto en los que las líneas contienen varios campos, separados por un carácter delimitador. Los archivos de registro contienen una entrada por línea y tienen el formato de campos separados por delimitadores, por lo que AWK es una excelente herramienta para la normalización.

Analice el archivo **applicationX_in_epoch.log** que se muestra a continuación. La fuente del archivo de registro no es relevante.

```
2|Z|1219071600|AF|0
3|N|1219158000|AF|89
4|N|1220799600|AS|12
1|Z|1220886000|AS|67
5|N|1220972400|EU|23
6|R|1221058800|OC|89
```

El archivo de registro anterior fue generado por la aplicación X. Los aspectos relevantes del archivo son los siguientes:

- Las columnas están separadas, o delimitadas, por el carácter **|** Por lo tanto, los datos tienen cinco columnas.
- La tercera columna contiene las marcas de hora en Unix Epoch.
- El archivo tiene una línea adicional al final. Esto será importante más adelante en la práctica de laboratorio.

Suponga que un analista de archivos de registro necesita convertir las marcas de hora al formato legible. Siga los pasos que se detallan a continuación y utilice AWK para realizar la conversión manual fácilmente:

a. Abra la **VM CyberOps Workstation** y, luego, abra una ventana del terminal.

b. Utilice el comando **cd** para pasar al directorio **/home/analyst/lab.support.files/**. Allí se almacena una copia del archivo que se muestra arriba.

```
[analyst@secOps ~]$ cd /home/analyst/lab.support.files/
[analyst@secOps lab.support.files]$ ls -l
total 580
-rw-r--r-- 1 analyst analyst 649 Jun 28 18:34 apache_in_epoch.log
-rw-r--r-- 1 analyst analyst 126 Jun 28 11:13 applicationX_in_epoch.log
drwxr-xr-x 4 analyst analyst 4096 Aug 7 15:29 attack_scripts
-rw-r--r-- 1 analyst analyst 102 Jul 20 09:37 confidential.txt
<output omitted>
[analyst@secOps lab.support.files]$
```

c. Emita el siguiente comando de AWK para convertir e imprimir el resultado en el terminal:

Nota: La flecha arriba se puede utilizar para editar los errores de escritura en la entrada del comando anterior.

```
[analyst@secOps lab.support.files]$ awk 'BEGIN {FS=OFS="|"}
{$3=strftime("%c",$3)} {print}' applicationX_in_epoch.log
2|Z|Mon 18 Aug 2008 11:00:00 AM EDT|AF|0
3|N|Tue 19 Aug 2008 11:00:00 AM EDT|AF|89
4|N|Sun 07 Sep 2008 11:00:00 AM EDT|AS|12
1|Z|Mon 08 Sep 2008 11:00:00 AM EDT|AS|67
5|N|Tue 09 Sep 2008 11:00:00 AM EDT|EU|23
6|R|Wed 10 Sep 2008 11:00:00 AM EDT|OC|89
||Wed 31 Dec 1969 07:00:00 PM EST
[analyst@secOps lab.support.files]$
```

El comando anterior es un script de AWK. Puede parecer complicado. La estructura principal del script de AWK es la siguiente:

- **awk**: invoca al intérprete de AWK.
- **'BEGIN**: define el comienzo del script.
- **{}**: define las acciones que se deben realizar en cada línea del archivo de texto de entrada. Un script de AWK puede tener varias acciones.
- **FS = OFS = "|"**: esto define el separador de campos (es decir, el delimitador) como el símbolo (|) de la barra. En distintos archivos de texto se pueden utilizar diferentes caracteres delimitadores para separar campos. Este operador permite que el usuario defina qué carácter se utiliza como el separador de campos en el archivo de texto actual.
- **$3**: esto hace referencia al valor presente en la tercera columna de la línea actual. En el archivo **applicationX_in_epoch.log**, la tercera columna contiene la marca de hora en Epoch que se tiene que convertir.
- **strftime**: esta es la función interna de AWK diseñada para trabajar con la hora. Los valores %c y $3 entre paréntesis son los parámetros que se pasan a **strftime**.
- **applicationX_in_epoch.log**: este es el archivo de texto de entrada que se tiene que cargar y utilizar. Como ya está en el directorio **lab.support.files**, no es necesario que agregue la información de la ruta: **/home/analyst/lab.support.files/applicationX_in_epoch.log**.

La primera acción del script, definida en el primer conjunto de llaves es definir el carácter separador de campos como la barra vertical ("|"). Luego, en el segundo conjunto de llaves, se reescribe la tercera columna de cada línea con el resultado de la función **strftime()** que se ejecutó. **strftime()** es una función interna de AWK creada para manejar la conversión de la hora. Observe que el script le ordena a la función que utilice el contenido de la tercera columna de cada línea antes del cambio (**$3**) y que dé formato a la salida (**%c**).

¿Se convirtieron las marcas de hora Unix Epoch al formato legible? ¿Se modificaron los otros campos? Explique.

Comparen el contenido del archivo con la salida impresa. ¿Por qué está la siguiente línea: **||Wed 31 Dec 1969 07:00:00 PM EST**?

d. Utilice **nano** (o el editor de texto que prefiera) para quitar la línea vacía adicional del final del archivo y vuelva a ejecutar el script **AWK** mediante el uso de la flecha arriba para encontrarlo en el búfer del historial de comandos.

```
[analyst@secOps lab.support.files]$ nano applicationX_in_epoch.log
```

¿Ahora la salida es correcta? Explique.

e. Si bien imprimir el resultado en la pantalla es útil para solucionar problemas en el script, los analistas probablemente necesitarán guardar la salida en un archivo de texto. Redirija la salida del script anterior a un archivo de nombre **applicationX_in_human.log** para guardarlo en un archivo:

```
[analyst@secOps lab.support.files]$ awk 'BEGIN {FS=OFS="|"}
{$3=strftime("%c",$3)} {print}' applicationX_in_epoch.log >
applicationX_in_human.log
[analyst@secOps lab.support.files]$
```

¿Qué imprimió el comando anterior? ¿Es esperable?

f. Utilice **cat** para ver el archivo **applicationX_in_human.log**. Observe que ahora se quitó la línea adicional y que las marcas de hora correspondientes a las entradas de registro se han convertido al formato legible.

```
[analyst@secOps lab.support.files]$ cat applicationX_in_human.log
2|Z|Mon 18 Aug 2008 11:00:00 AM EDT|AF|0
3|N|Tue 19 Aug 2008 11:00:00 AM EDT|AF|89
4|N|Sun 07 Sep 2008 11:00:00 AM EDT|AS|12
1|Z|Mon 08 Sep 2008 11:00:00 AM EDT|AS|67
5|N|Tue 09 Sep 2008 11:00:00 AM EDT|EU|23
6|R|Wed 10 Sep 2008 11:00:00 AM EDT|OC|89
[analyst@secOps lab.support.files]$
```

Parte 2: Normalizar marcas de hora en un archivo de registro Apache

En forma similar a lo que se hizo con el archivo **applicationX_in_epoch.log** , los archivos de registro Apache también se pueden normalizar. Siga los pasos que se detallan a continuación para convertir marcas de hora Unix Epoch al formato legible. Analicen el siguiente archivo de registro Apache, **apache_in_epoch.log**:

```
[analyst@secOps lab.support.files]$ cat apache_in_epoch.log
```

```
198.51.100.213 - - [1219071600] "GET
/twiki/bin/edit/Main/Double_bounce_sender?topicparent=Main.ConfigurationVariables
HTTP/1.1" 401 12846
198.51.100.213 - - [1219158000] "GET
/twiki/bin/rdiff/TWiki/NewUserTemplate?rev1=1.3&rev2=1.2 HTTP/1.1" 200 4523
198.51.100.213 - - [1220799600] "GET /mailman/listinfo/hsdivision HTTP/1.1" 200 6291
198.51.100.213 - - [1220886000] "GET /twiki/bin/view/TWiki/WikiSyntax HTTP/1.1" 200
7352
198.51.100.213 - - [1220972400] "GET /twiki/bin/view/Main/DCCAndPostFix HTTP/1.1" 200
5253
198.51.100.213 - - [1221058800] "GET
/twiki/bin/oops/TWiki/AppendixFileSystem?template=oopsmore&m1=1.12&m2=1.12 HTTP/1.1"
200 11382
```

El archivo de registro Apache anterior contiene seis entradas que registran eventos relacionados con el servidor web Apache. Cada entrada tiene siete campos. Los campos están delimitados por un espacio:

- La primera columna contiene la dirección IPv4 (**198.51.100.213**) del cliente web que está elevando la solicitud.
- La segunda y tercera columna no se utilizan, y se emplea un carácter de "**-**" para representar que no hay ningún valor.
- La cuarta columna contiene la marca de hora en Unix Epoch, por ejemplo: **[1219071600]**.
- La quinta columna contiene texto con detalles sobre el evento; eso incluye direcciones URL y los parámetros de la petición web. Las seis entradas son mensajes HTTP GET. Como estos mensajes incluyen espacios, todo el campo está encerrado entre comillas.
- La sexta columna contiene el código de estado HTTP, por ejemplo: **401**.
- La séptima columna contiene el tamaño de la respuesta al cliente (en bytes), por ejemplo: **12846**.

En forma similar a la Parte 1, se creará un script para convertir la marca de hora del formato Epoch al legible.

a. En primer lugar, responda las siguientes preguntas. Son cruciales para la construcción del script.

   En el contexto de la conversión de marcas de hora, ¿qué carácter funcionará bien como carácter delimitador para el archivo de registro Apache anterior?

   ¿Cuántas columnas contiene el archivo de registro Apache anterior?

   ¿En el archivo de registro Apache anterior, qué columna contiene la marca de hora en Unix Epoch?

b. En el terminal de la **VM CyberOps Workstation** se almacena una copia del archivo de registro Apache (apache_in_epoch.log), en /home/analyst/lab.support.files.

c. Utilice un script de **awk** para convertir el campo de la marca de hora al formato legible. Observe que el comando contiene el mismo script que ya se utilizó, pero con algunos ajustes en el campo de la marca de hora y en el nombre de archivo.

```
[analyst@secOps lab.support.files]$ awk 'BEGIN {FS=OFS=" "}
{$4=strftime("%c",$4)} {print}' apache_in_epoch.log
```

¿El script pudo convertir las marcas de hora correctamente? Describa la salida.

d. Antes de seguir adelante, piense sobre la salida del script.

¿Puede decir qué hizo que la salida fuese incorrecta? ¿El script es incorrecto? ¿Cuáles son las diferencias relevantes entre **applicationX_in_epoch.log** y **apache_in_epoch.log**?

e. Para corregir el problema se deben quitar los corchetes del campo de la marca de hora antes de realizar la conversión. Corrija el script; para ello, agregue dos acciones antes de la conversión, de la siguiente manera:

```
[analyst@secOps lab.support.files]$ awk 'BEGIN {FS=OFS=" "}
{gsub(/\[|\]/,"",$4)}{print}{$4=strftime("%c",$4)}{print}'
apache_in_epoch.log
```

Observe que, después de especificar el espacio como delimitador con **{FS=OFS=" "}**, hay una acción de expresión regular para hacer coincidir y reemplazar los corchetes por una cadena vacía; eso quita efectivamente los corchetes que aparecen en el campo de la marca de hora. La segunda acción imprime la línea actualizada para que se pueda realizar la acción de la conversión.

- **gsub()**: esta es una función interna de AWK que se utiliza para ubicar y sustituir cadenas. En el script anterior, **gsub()** recibió tres parámetros separados por comas, que se describen a continuación.
- **/\[|\]/:** esta es una expresión regular que se pasa a **gsub()**como primer parámetro. La expresión regular debe leerse como '**find "[" OR "]"**'. A continuación se muestra el desglose de la expresión:
  - El primer y último carácter "/" marcan el comienzo y el final del bloque de búsqueda. Cualquier elemento que se incluya entre la primera y la segunda "/" estará relacionado con la búsqueda. El carácter "\" se utiliza para dar escape al siguiente "[". El escape es necesario porque "[" también puede ser utilizado por un operador en expresiones regulares. Al escapar el "[" con una barra "\" precedente, le estamos diciendo al intérprete que el "]" forma parte del contenido y no de un operador. El carácter "|" es el operador OR. Tengan en cuenta que la barra "|" no tiene escape y, por lo tanto puede verse como un operador. Por último, la expresión regular da escape al corchete de cierre con "\]", como se hizo antes.
- **""**: esto representa la ausencia de caracteres, o una cadena vacía. Este parámetro le dice a **gsub()** con qué debe reemplazar "[" y "]", cuando los encuentre. Al reemplazar "[" y "]" por "", **gsub()** quita efectivamente los caracteres "[" y "]".
- **$4**: esto le dice a **gsub()** que trabaje solamente en la cuarta columna de la línea actual, la columna de la marca de hora.

Nota: La interpretación de expresiones regulares es un tema del examen de SECOPS. Las expresiones regulares se analizan más detalladamente en otra práctica de laboratorio de este capítulo. Sin embargo, pueden buscar tutoriales en Internet.

f. En un terminal de la VM CyberOps Workstation, ejecute el script modificado, de la siguiente manera:

```
[analyst@secOps lab.support.files]$ awk 'BEGIN {FS=OFS=" "}
{gsub(/\[|\]/,"",$4)}{print}{$4=strftime("%c",$4)}{print}'
apache_in_epoch.log
```

¿El script pudo convertir las marcas de hora correctamente esta vez? Describa la salida.

g. Apague CyberOps Workstation VM si lo desea.

Parte 3: Preparación de archivos de registro en Security Onion

Como la normalización de los archivos de registro es importante, las herramientas para el análisis de registros a menudo incluyen características de normalización de registros. Las herramientas que no incluyen tales características a menudo dependen de complementos para la normalización y preparación de registros. La meta de estos complementos es permitir que las herramientas para el análisis de registros normalicen y preparen los archivos de registro recibidos para que las herramientas puedan utilizarlos.

El dispositivo Security Onion depende de diversas herramientas para proporcionar servicios de análisis de registros. **ELK**, **Zeek**, **Snort** and **SGUIL** sin duda son las más utilizadas.

ELK (Elasticsearch, Logstash y Kibana) es una solución para lograr lo siguiente:

- Normalizar, almacenar e indexar registros en volúmenes y velocidades sin límite.
- Proporcionar una interfaz de búsqueda y API simple y transparente.
- Proporcionar una infraestructura para alertar, informar y compartir registros.
- Contar con un sistema complementario para realizar acciones con los registros.
- Funcionar como un proyecto totalmente gratuito y de código abierto.

Zeek (formalmente llamado Bro)es un marco diseñado para analizar el tráfico de red y generar registros de eventos en función de dicho tráfico. Al analizar el tráfico de red, Zeek crea registros que describen eventos como los siguientes:

- Conexiones de red TCP/UDP/ICMP
- Actividad del DNS
- Actividad de FTP
- Peticiones y respuestas HTTPS
- Protocolos de enlace SSL/TLS

Snort y SGUIL

Snort es un IDS que depende de reglas predefinidas para señalar tráfico potencialmente dañino. Snort analiza todas las porciones de los paquetes de red (encabezados y cargas útiles) en busca de los patrones definidos en sus reglas. Cuando los encuentra, Snort toma la medida definida en la misma regla.

SGUIL proporciona una interfaz gráfica para los registros y las alertas de Snort, lo que permite que el analista de seguridad pase de SGUIL a otras herramientas para obtener más información. Por ejemplo: si se envía un paquete potencialmente malicioso al servidor web de la organización y Snort elevó una alerta al respecto, SGUIL incluirá esa alerta en una lista. Entonces, el analista puede hacer clic derecho sobre esa alerta para buscar en las bases de datos de ELSA o Bro y así comprender mejor el evento.

Nota: El listado de directorios puede diferir del resultado de muestra que se incluye a continuación.

Paso 1: Inice la máquina virtual Security Onion

Inicie la **VM de Security Onion** desde el panel de control de VirtualBox (nombre de usuario: **analyst**/contraseña: **cyberops**).

Paso 2: Registro de Zeek en Security Onion

a. Abra una ventana de terminal en la VM Security Onion. Haga clic derecho en el escritorio. En el menú emergente, seleccione **Abrir terminal**.

b. Los registros de Zeek se almacenan en **/nsm/bro/logs/**. Como es habitual en sistemas Linux, los archivos de registro se rotan en función de la fecha, se les cambia el nombre y se almacenan en el disco. Los archivos de registro actuales se pueden encontrar en el directorio current. Desde la ventana del terminal, cambien de directorio con el siguiente comando:

```
analyst@SecOnion:~$ cd /nsm/bro/logs/current
analyst@SecOnion:/nsm/logs/current$
```

c. Utilice el comando **ls -l** para ver todos los archivos de registro que generó Zeek:

Nota: Depende del estado de la máquina virtual, es posible que aún no haya archivos de registro.

Paso 3: Archivos de registro de Snort

a. Los archivos de registro de Snort se pueden encontrar en **/nsm/sensor_data/**. Cambie de directorio de la siguiente manera:

```
analyst@SecOnion:/nsm/bro/logs/current$ cd /nsm/sensor_data
analyst@SecOnion:/nsm/sensor_data$
```

b. Utilice el comando ls -l para ver todos los archivos de registro que generó Snort:

```
analyst@SecOnion:/nsm/sensor_data$ ls -l
total 12
drwxrwxr-x 7 sguil sguil 4096 Jun 19 18:09 seconion-eth0
drwxrwxr-x 5 sguil sguil 4096 Jun 19 18:09 seconion-eth1
drwxrwxr-x 7 sguil sguil 4096 Jun 19 18:32 seconion-import
```

c. Observe que Security Onion separa los archivos en función de la interfaz. Debido a que la imagen **VM de Security Onion** tiene dos interfaces configuradas como sensores y una carpeta especial para los datos importados, se conservan tres directorios. Utilice el comando **ls –l seconion-eth0** para ver los archivos que generó la interfaz ethernet 0.

```
analyst@SecOnion:/nsm/sensor_data$ ls -l seconion-eth0
total 28
drwxrwxr-x 2 sguil sguil 4096 Jun 19 18:09 argus
drwxrwxr-x 3 sguil sguil 4096 Jun 19 18:09 dailylogs
drwxrwxr-x 2 sguil sguil 4096 Jun 19 18:09 portscans
drwxrwxr-x 2 sguil sguil 4096 Jun 19 18:09 sancp
drwxr-xr-x 2 sguil sguil 4096 Jun 19 18:24 snort-1
-rw-r—r— 1 sguil sguil 5594 Jun 19 18:31 snort-1.stats
-rw-r--r-- 1 root root 0 Jun 19 18:09 snort.stats
```

Paso 4: Registros varios

a. Si bien en el directorio **/nsm/** se almacenan algunos archivos de registro, se pueden encontrar otros más específicos en **/var/log/nsm/**. Cambie de directorio y utilice el comando **ls** para ver todos los archivos de registro presentes en el directorio.

```
analyst@SecOnion:/nsm/sensor_data$ cd /var/log/nsm/
analyst@SecOnion:/var/log/nsm$ ls
eth0-packets.log sid_changes.log
netsniff-sync.log so-elastic-configure-kibana-dashboards.log
ossec_agent.log so-elasticsearch-pipelines.log
```

```
pulledpork.log so-sensor-backup-config.log
seconion-eth0 so-server-backup-config.log
importación secundaria sosetup.log
securityonion so-zeek-cron.log
sensor-clean.log squert-ip2c-5min.log
sensor-clean.log.1.gz squert-ip2c.log
sensor-clean.log.2.gz squert_update.log
sensor-newday-argus.log watchdog.log
sensor-newday-http-agent.log watchdog.log.1.gz
sensor-newday-pcap.log watchdog.log.2.gz
sguil-db-purge.log
```

Observe que el directorio que se muestra arriba también contiene registros utilizados por herramientas secundarias como **OSSEC** y **Squert**.

b. Los registros de ELK se pueden encontrar en el directorio **/var/log**. Cambie de directorio y utilice el comando **ls** para ver todos los archivos de registro presentes en el directorio.

```
analyst@SecOnion:/var/log/nsm$ cd ..
analista @SecOnion: /var/log$ ls
alternatives.log debug kern.log.1 samba
alternatives.log.1 debug.1 kern.log.2.gz sguild
apache2 debug.2.gz kibana so-boot.log
apt dmesg lastlog syslog
auth.log domain_stats lightdm syslog.1
auth.log.1 dpkg.log logstash syslog.2.gz
auth.log.2.gz dpkg.log.1 lpr.log syslog.3.gz
arranque elastalert mail.err syslog.4.gz
boot.log elasticsearch mail.info actualizaciones desatendidos
bootstrap.log error mail.log user.log
error.1 mail.warn user.log.1
btmp.1 error.2.gz mensajes user.log.2.gz
Mensajes de registro de error cron.log. 1 wtmp
cron.log.1 mensajes freq_server. 2.gz wtmp.1
cron.log.2.gz freq_server_dns mysql xorg.0.log
curador fsck nsm Xorg.0.log.old
daemon.log gpu-manager.log ntpstats
daemon.log.1 instalador redis
daemon.log.2.gz kern.log sal
```

c. Tómese un momento para investigar estas herramientas en Google y responda las siguientes preguntas:

En cada una de las herramientas antes enumeradas, describa su función, importancia y ubicación en el flujo de trabajo del analista de seguridad.

Reflexión

La normalización de registros es importante y depende del entorno implementado.

Las herramientas populares incluyen sus propias características de normalización, pero los registros también se pueden normalizar manualmente.

Cuando esté normalizando y preparando archivos de registro manualmente, revise bien los scripts para garantizar que se llegue al resultado deseado. Un script de normalización mal escrito puede modificar los datos, y eso afectaría directamente el trabajo del analista.