Detectando APT's

Sergio Rosello

Resumen

En esta practica se pretende hacer un sistema de detección de APT. El problema es que los APT están diseñados específicamente para ser silenciosos y no llamar la atención.

Instalación del entorno con Docker

Se ha descargado el archivo proporcionado en el foro y se ha ejecutado el siguiente comando:

```
systemctl start docker
```

A partir de este momento, se le cambia el nombre al archivo compose, para no tener que indicarle el nombre.

Apache2 (Entrada de datos)

Para obtener la información de los registros, he optado por iniciar un servidor apache2 sencillo y medir los registros de peticiones entrantes. Estos registros se guardan en el archivo `access_log` en el directorio `/var/log/httpd/`.

```
trizen -Syu apache
systemctl start httpd
```

Logstash (Consumo de datos)

Ahora que ya tenemos el dockerfile con la configuración de `elasticsearch` y `kibana`, vamos a crear una sección para logstash. De esta forma, conseguimos que toda nuestra infraestructura este dockerizada. Insertamos en el dockerfile el siguiente segmento de codigo.

```
logstash:
  image: docker.elastic.co/logstash/logstash:7.1.1
  container_name: logstash
  ports:
      - 5000:5000
      - 9600:9600
  networks:
      - docker-elk
  volumes:
      - type: bind
        source: /var/log/httpd/access_log
        target: /var/log/httpd/access_log
      - type: bind
```

```
        source: ./logstash/pipelines/
        target: /usr/share/logstash/pipeline
  depends_on:
      -  elasticsearch
```

Creamos el archivo de configuración del logstash. Este es el archivo que le dice a logstash de donde ingiere los datos, como los trata y donde los manda.

```
input {
     file {
         path => "/var/log/httpd/access_log"
         start_position => beginning
         type => "apache-access"
     }
 }

 filter {
    if [type] == "apache-access" {
        grok {
          match => { "message" => "%{COMBINEDAPACHELOG}" }
        }
    }
    date {
        match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]
    }
 }

 output {
     elasticsearch {
         hosts => [ "elasticsearch:9200" ]
     }
 }
```

Esto lo he indicado en el dockerfile en la linea `source: /var/log/httpd/access_log`. La siguiente linea en el archivo indica a la instancia docker que genere el mismo contenido del archivo anterior en el mismo directorio en la maquina. Realmente, no es necesario que tengan la misma ruta, pero por simplicidad, lo he indicado así.

Dockerfile completo

Al terminar la configuración, queda un dockerfile como el mostrado a continuación.

```
version: '3.7'
services:
     elasticsearch:
       container_name: elasticsearch
```

```
    image: docker.elastic.co/elasticsearch/elasticsearch:7.1.1
    environment:
      - 'discovery.type=single-node'
      - 'ES_JAVA_OPTS=-Xms256M -Xmx256M'
      - 'xpack.license.self_generated.type=trial'
    ports:
      - 9200:9200
      - 9300:9300
    networks:
      - docker-elk
  logstash:
     image: docker.elastic.co/logstash/logstash:7.1.1
     container_name: logstash
     ports:
         - 5000:5000
         - 9600:9600
     networks:
         - docker-elk
     volumes:
         - type: bind
           source: /var/log/httpd/access_log
           target: /var/log/httpd/access_log
         - type: bind
           source: ./logstash/pipelines/
           target: /usr/share/logstash/pipeline
     depends_on:
         -  elasticsearch
  kibana:
    container_name: kibana
    image: docker.elastic.co/kibana/kibana:7.1.1
    ports:
      - 5601:5601
    environment:
      - 'ELASTICSEARCH_HOSTS=http://elasticsearch:9200'
    networks:
      - docker-elk
    depends_on:
      - elasticsearch
networks:
  docker-elk:
    driver: bridge
```

Al acabar esta configuración, se inicia docker con el comando: `docker-compose up -d`, esperamos a que se inicien todos los servicios y empezamos a generar accesos al servidor. Ahora, `logstash` lee los datos de entrada del log en mi sistema y kibana puede interpretar los datos en elasticsearch de forma gráfica

para presentármelos.

Configuración de Kibana

Entramos en el portal, lo primero que aparece es un verificador de datos. Verificamos que hay datos en Kibana y configuramos para que nos muestre los datos del registro. Estos datos, se pueden visualizar ahora, de forma gráfica e intuitiva.

En este momento, cuando tengamos accesos maliciosos, o un APT, podremos detectarlo porque destaca ante los datos.

El propósito de Kibana es poder gestionar de forma sencilla la cantidad de datos que genera un sistema en producción como un servidor, sistema, o cualquier programa que genere registros.

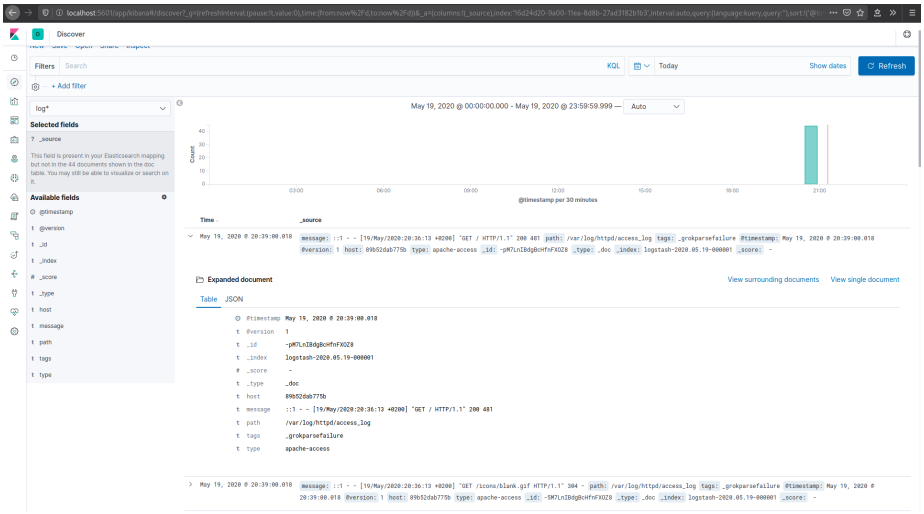

Figure 1: Kibana

En la imagen superior se muestra como se pueden acceder a los datos de forma visual e intuitiva. Especificamente, es un registro de acceso al servidor apache2 desde mi propia maquina.

Con las versiones de `kibana`, `logstash` y `elasticsearch` que se han instalado en el dockerfile, no hay necesidad de instalar de nuevo `x-pack` puesto a que en las tres viene ya instalado por defecto. Usando herramientas como `x-pack` de machine learning se le añade una capa mas de análisis a una herramienta ya muy útil.

Es una lastima no haber encentrado un dataset de APT compatible con esta configuración, pero a simple vista, se ve el potencial de la herramienta.