Laboratório - Converta dados em um formato universal

Objetivos

Parte 1: Normalizar Carimbos de Data/Hora em um Arquivo de Log

Parte 2: Normalizar Carimbos de Data/Hora em um Arquivo de Log Apache

Parte 3: Preparação do Arquivo de Log na Máquina Virtual Security Onion

Histórico/Cenário

Este laboratório irá prepará-lo para aprender onde os arquivos de log estão localizados e como manipular e visualizar os arquivos de log. **Entradas de Log** são gerados por dispositivos de rede, sistemas operacionais, aplicativos e vários tipos de dispositivos programáveis. Um arquivo contendo um fluxo sequenciado de tempo de entradas de registro é chamado de arquivo de **log**.

Por natureza, os arquivos de log registram eventos que são relevantes para a fonte. A sintaxe e o formato dos dados nas mensagens de log geralmente são definidos pelo desenvolvedor do aplicativo.

Portanto, a terminologia usada nas entradas de log geralmente varia de fonte para fonte. Por exemplo, dependendo da fonte, os termos login, logon, evento de autenticação e conexão do usuário podem aparecer nas entradas de log para descrever uma autenticação de usuário bem-sucedida em um servidor.

Muitas vezes, é desejável ter uma terminologia consistente e uniforme nos logs gerados por fontes diferentes. Isso é especialmente verdadeiro quando todos os arquivos de log estão sendo coletados por um ponto centralizado.

O termo **normalização** refere-se ao processo de conversão de partes de uma mensagem, neste caso uma entrada de log, em um formato comum.

Neste laboratório, você usará ferramentas de linha de comando para normalizar manualmente as entradas de registro. Na Parte 2, o campo de carimbo de data/hora será normalizado. Na Parte 3, o campo IPv6 será normalizado.

Nota: Embora existam vários plug-ins para realizar a normalização de log, é importante entender os fundamentos do processo de normalização.

Recursos necessários

- Máquina Virtual CyberOps Workstation
- Máquina Virtual Security Onion

Instruções

Parte 1: Normalizar Carimbos de Data/Hora em um Arquivo de Log

Os carimbos de data/hora são usados em entradas de log para especificar quando o evento registrado ocorreu. Embora seja uma prática recomendada registrar carimbos de data/hora em UTC, o formato do carimbo de data / hora varia de origem de log para origem de log. Existem dois formatos de carimbo de data/hora comuns, conhecidos como Unix Epoch e Human Readable.

Os timestamps Unix Epoch registram o tempo medindo o número de segundos que se passaram desde 1º de janeiro de 1970.

Os carimbos de data / hora legíveis por humanos registram o tempo, representando valores separados para ano, mês, dia, hora, minuto e segundo.

A data **Wed, 28 Jun 2017 13:27:19 GMT** legível por humanos em timestamp é o mesmo que **1498656439** em Unix Epoch.

Do ponto de vista da programação, é muito mais fácil trabalhar com o Epoch, pois permite operações de adição e subtração mais fáceis. De uma perspectiva de análise; no entanto, carimbos de data/hora legíveis por humanos são muito mais fáceis de interpretar.

Convertendo Epoch em Timestamps legíveis por humanos com AWK

AWK é uma linguagem de programação projetada para manipular arquivos de texto. É muito poderoso e especialmente útil ao lidar com arquivos de texto onde as linhas contêm vários campos, separados por um caractere delimitador. Os arquivos de log contêm uma entrada por linha e são formatados como campos separados por delimitadores, tornando o AWK uma ótima ferramenta para normalização.

Considere o arquivo **applicationX_in_epoch.log** abaixo. A origem do arquivo de log não é 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
```

O arquivo de log acima foi gerado pelo que chamaremos de aplicativo X. Os aspectos relevantes do arquivo são:

- As colunas são separadas ou delimitadas pelo caractere **|**. Portanto, os dados têm cinco colunas.
- A terceira coluna contém carimbos de data/hora no Unix Epoch.
- O arquivo possui uma linha extra no final. Isso será importante posteriormente no laboratório.

Suponha que um analista de log precise converter os carimbos de data/hora em um formato legível. Siga as etapas abaixo para usar o AWK para realizar facilmente a conversão manual:

a. Inicie a **VM CyberOps Workstation** e, em seguida, abra uma janela de terminal.

b. Use o comando **cd** para mudar para o diretório **/home/analyst/lab.support.files/**. Uma cópia do arquivo mostrado acima é armazenada lá.

```
[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>
[analista @secOps lab.support.files] $
```

c. Emita o seguinte comando AWK para converter e imprimir o resultado no terminal:

Nota: A seta para cima pode ser usada para editar os erros de digitação na entrada do 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]$
```

O comando acima é um script AWK. Pode parecer complicado. A estrutura principal do script AWK acima é a seguinte:

- **awk** – Isso invoca o interpretador AWK.
- **'BEGIN** – Isso define o início do script.
- **{}** – Isso define as ações a serem executadas em cada linha do arquivo de texto de entrada. Um script AWK pode ter várias ações.
- **FS = OFS = "|" –** Isso define o separador de campo (ou seja, delimitador) como o símbolo de barra (|). Arquivos de texto diferentes podem usar caracteres delimitadores diferentes para campos separados. Este operador permite ao usuário definir qual caractere é usado como separador de campo no arquivo de texto atual.
- **$3 –** Isso se refere ao valor na terceira coluna da linha atual. No **applicationX_in_epoch.log**, a terceira coluna contém o carimbo de data/hora na época a ser convertido.
- **strftime -** Esta é uma função interna AWK projetada para funcionar com o tempo. O %c e $3 entre parênteses estão os parâmetros passados para **strftime**.
- **applicationX_in_epoch.log –** Este é o arquivo de texto de entrada a ser carregado e usado. Devido já estar no diretório **lab.support.files**, você não precisa adicionar informações de caminho, **/home/analyst/lab.support.files/applicationX_in_epoch.log**.

A primeira ação do script definida no primeiro conjunto de chaves é definir o caractere separador de campo como o "|". Então, no segundo conjunto de chaves, ele reescreve a terceira coluna de cada linha com o resultado da execução da função **strftime()**. **strftime()** é uma função AWK interna criada para lidar com a conversão de tempo. Observe que o script diz à função para usar o conteúdo da terceira coluna de cada linha antes da mudança (**$3**) e formatar a saída (**%c**).

Os carimbos de data/hora Unix Epoch foram convertidos para o formato legível por humanos? Os outros campos foram modificados? Explique.

Compare o conteúdo do arquivo e a saída impressa. Por que existe a linha, **||Wed 31 Dec 1969 07:00:00 PM EST**?

d. Use o **nano** (ou seu editor de texto favorito) para remover a linha vazia extra no final do arquivo e executar o script **AWK** novamente usando a seta para cima para localizá-lo no buffer do histórico de comandos.

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

A saída está correta agora? Explique.

e. Embora imprimir o resultado na tela seja útil para solucionar problemas do script, os analistas provavelmente precisarão salvar a saída em um arquivo de texto. Redirecione a saída do script acima para um arquivo chamado **applicationX_in_human.log** para salvá-lo em um arquivo:

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

O que foi impresso pelo comando acima? Isso é esperado?

f. Use **cat** para ver o **applicationX_in_human.log**. Observe que a linha extra foi removida e os carimbos de data / hora para as entradas de registro foram convertidos para o formato legível por humanos.

```
[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 carimbos de data/hora em um arquivo de registro Apache

Semelhante ao que foi feito com o arquivo **applicationX_in_epoch.log**, Os arquivos de log do servidor da web Apache também podem ser normalizados. Siga as etapas abaixo para converter Unix Epoch em carimbos de data/hora legíveis por humanos. Considere o seguinte arquivo de log do 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
```

O arquivo Apache Log acima contém seis entradas que registram eventos relacionados ao servidor web Apache. Cada entrada possui sete campos. Os campos são delimitados por um espaço:

- A primeira coluna contém o endereço IPv4, **198.51.100.213**, do cliente da web que faz a solicitação.
- A segunda e terceira colunas não são usadas e um caractere "**-**" é usado para representar nenhum valor.
- A quarta coluna contém o carimbo de data/hora no tempo Unix Epoch, por exemplo **[1219071600]**.
- A quinta coluna contém texto com detalhes sobre o evento, incluindo URLs e parâmetros de solicitação da web. Todas as seis entradas são mensagens HTTP GET. Como essas mensagens incluem espaços, todo o campo é colocado entre aspas.
- A sexta coluna contém o código de status HTTP, por exemplo **401**.
- A sétima coluna contém o tamanho da resposta ao cliente (em bytes), por exemplo **12846**.

Como na Parte 1, um script será criado para converter o carimbo de data / hora de Epoch em Human Readable.

a. Primeiro, responda às perguntas abaixo. Eles são fundamentais para a construção do roteiro.

   No contexto da conversão de carimbo de data/hora, qual caractere funcionaria como um bom caractere delimitador para o arquivo de log do Apache acima?

   Quantas colunas o arquivo de log do Apache contém?

   No arquivo de log do Apache acima, qual coluna contém o carimbo de data/hora Unix Epoch?

b. No terminal da **VM CyberOps Workstation**,uma cópia do arquivo de log do Apache, apache_in_epoch.log, é armazenada em /home/analyst/lab.support.files.

c. Use um script **awk** para converter o campo de carimbo de data/hora em um formato legível por humanos. Observe que o comando contém o mesmo script usado anteriormente, mas com alguns ajustes para o delimitador, campo de carimbo de data/hora e nome do arquivo.

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

   O script conseguiu converter corretamente os carimbos de data/hora? Descreva a saída.

d. Antes de prosseguir, pense sobre a saída do script.

   Você consegue adivinhar o que causou a saída incorreta? O script está incorreto? Quais são as diferenças relevantes entre **applicationX_in_epoch.log** e **apache_in_epoch.log**?

e. Para corrigir o problema, os colchetes devem ser removidos do campo de carimbo de data / hora antes que a conversão ocorra. Ajuste o script adicionando duas ações antes da conversão, conforme mostrado abaixo:

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

Observe depois de especificar o espaço como delimitador com **{FS=OFS=" "}**, há uma ação de expressão regular para corresponder e substituir os colchetes por uma string vazia, removendo efetivamente os colchetes que aparecem no campo de carimbo de data/hora. A segunda ação imprime a linha atualizada para que a ação de conversão possa ser realizada.

- **gsub() –** Esta é uma função AWK interna usada para localizar e substituir strings. No roteiro acima, **gsub()** recebeu três parâmetros separados por vírgula, descritos abaixo.
- **/\[|\]/ –** Esta é uma expressão regular passada para **gsub()** como o primeiro parâmetro. A expressão regular deve ser lida como '**find "[" OR "]"**'. Abaixo está o detalhamento da expressão:
  - O primeiro e o último caractere "/" marcam o início e o fim do bloco de pesquisa. Qualquer coisa entre o primeiro "/" e o segundo "/" está relacionado à pesquisa. O caractere "\" é usado para escapar do seguinte "[". O escape é necessário porque "[" também pode ser usado por um operador em expressões regulares. Ao escapar do "[" com um "\" inicial, dizemos ao intérprete que o "]" é parte do conteúdo e não um operador. O "|" caractere é o operador OR. Observe que o "|" não tem escape e, portanto, será visto como um operador. Por último, a expressão regular escapa o colchete de fechamento com "\]", como feito antes.
- **""** – Isso representa nenhum caractere ou uma string vazia. Este parâmetro diz ao **gsub()** o que substituir o "[" e "]", quando encontrado. Ao substituir o "[" e "]" por "", **gsub()** remove efetivamente os caracteres "[" e "]".
- **$4** – Isso diz **gsub()** para funcionar apenas na quarta coluna da linha atual, a coluna de carimbo de data / hora.

Nota: A interpretação da expressão regular é um tópico do exame SECOPS. As expressões regulares são abordadas com mais detalhes em outro laboratório neste capítulo. No entanto, você pode querer pesquisar na Internet por tutoriais.

f. Em um terminal VM CyberOps Workstation, execute o script ajustado, da seguinte maneira:

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

O script conseguiu converter corretamente os carimbos de data/hora desta vez? Descreva a saída.

g. Desligue o CyberOps Workstation VM se desejar.

Parte 3: Preparação do arquivo de log na máquina virtual Security Onion

Como a normalização do arquivo de log é importante, as ferramentas de análise de log geralmente incluem recursos de normalização de log. As ferramentas que não incluem esses recursos geralmente dependem de plug-ins para normalização e preparação de log. O objetivo desses plug-ins é permitir que as ferramentas de análise de log normalizem e preparem os arquivos de log recebidos para consumo da ferramenta.

O dispositivo Security Onion conta com várias ferramentas para fornecer serviços de análise de log. **ELK**, **Zeek**, **Snort** e **SGUIL** são indiscutivelmente as ferramentas mais usadas.

ELK (Elasticsearch, Logstash e Kibana) é uma solução para alcançar o seguinte:

- Normalize, armazene e indexe logs em taxas e volumes ilimitados.
- Fornece uma interface de pesquisa simples e limpa e API.
- Fornece uma infraestrutura para alertar, relatar e compartilhar logs.
- Sistema de plug-in para realizar ações com logs.
- Exista como um projeto totalmente gratuito e de código aberto.

Zeek(anteriormente chamado de Bro) é uma estrutura projetada para analisar o tráfego de rede passivamente e gerar logs de eventos com base nele. Após a análise do tráfego de rede, Zeek cria logs que descrevem eventos como o seguinte:

- Conexões de rede TCP/UDP/ICMP
- Atividade DNS
- Atividade de FTP
- Solicitações e respostas HTTPS
- SSL/TLS handshakes

Snort e SGUIL

Snort é um IDS que se baseia em regras predefinidas para sinalizar tráfego potencialmente prejudicial. O Snort examina todas as partes dos pacotes de rede (cabeçalhos e carga útil), procurando padrões definidos em suas regras. Quando encontrado, o Snort executa a ação definida na mesma regra.

O SGUIL fornece uma interface gráfica para logs e alertas do Snort, permitindo que um analista de segurança pivote do SGUIL em outras ferramentas para obter mais informações. Por exemplo, se um pacote potencialmente malicioso for enviado ao servidor web da organização e o Snort disparar um alerta sobre isso, o SGUIL listará esse alerta. O analista pode clicar com o botão direito nesse alerta para pesquisar os bancos de dados ELSA ou Bro para uma melhor compreensão do evento.

Nota: A lista de diretórios pode ser diferente da saída de amostra mostrada abaixo.

Etapa 1: Inicie o Security Onion VM.

Lançar a **VM Security Onion** do painel do VirtualBox (username: **analyst** / password: **cyberops**).

Etapa 2: Zeek registra no Security Onion

a. Abra uma janela de terminal no Security Onion VM. Clique com o botão direito na área de trabalho. No menu pop-up, selecione**Open Terminal**.

b. Os registros Zeek são armazenados em **/nsm/bro/logs/**. Como de costume nos sistemas Linux, os arquivos de log são alternados com base na data, renomeados e armazenados no disco. Os arquivos de log atuais podem ser encontrados no diretório atual. Na janela do terminal, altere o diretório usando o seguinte comando.

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

c. Use o comando **ls -l** para ver os arquivos de log gerados pelo Zeek:

Nota: Depende do estado da máquina virtual, pode não haver nenhum arquivo de log ainda.

Etapa 3: Snort Logs

a. Os logs do Snort podem ser encontrados em **/nsm/sensor_data/**. Altere o diretório conforme a seguir.

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

b. Use o comando ls -l para ver todos os arquivos de log gerados pelo 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 o Security Onion separa os arquivos com base na interface. Devido a imagem da **VM Security Onion** possuir duas interfaces configuradas como sensores e uma pasta especial para os dados importados, sendo mantidos três diretórios. User o comando **ls –l seconion-eth0** para ver os arquivos gerados pela interface eth0.

```
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
```

Etapa 4: Vários Logs

a. Enquanto o diretório **/nsm/** armazena alguns arquivos de log, arquivos de log mais específicos podem ser encontrados em **/var/log/nsm/**. Mude o diretório e use o comando **ls** para ver todos os arquivos de log no diretório.

```
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
seconion-import 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 o diretório mostrado acima também contém logs usados por ferramentas secundárias, como **OSSEC** e **Squert**.

b. Os logs ELK podem ser encontrados no diretório **/var/log**. Mude o diretório e use o comando**ls** para listar os arquivos e diretórios.

```
analyst@SecOnion:/var/log/nsm$ cd ..
analyst@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
boot elastalert mail.err syslog.4.gz
boot.log elasticsearch mail.info unattended-upgrades
bootstrap.log error mail.log user.log
btmp error.1 mail.warn user.log.1
btmp.1 error.2.gz messages user.log.2.gz
cron.log faillog messages.1 wtmp
cron.log.1 freq_server messages.2.gz wtmp.1
cron.log.2.gz freq_server_dns mysql Xorg.0.log
curator fsck nsm Xorg.0.log.old
daemon.log gpu-manager.log ntpstats
daemon.log.1 installer redis
daemon.log.2.gz kern.log salt
```

c. Reserve um tempo para pesquisar essas ferramentas secundárias no Google e responder às perguntas abaixo:

Para cada uma das ferramentas listadas acima, descreva a função, importância e posicionamento no fluxo de trabalho do analista de segurança.

Reflexão

A normalização do log é importante e depende do ambiente implementado.

As ferramentas populares incluem seus próprios recursos de normalização, mas a normalização de log também pode ser feita manualmente.

Ao normalizar e preparar manualmente os arquivos de log, verifique novamente os scripts para garantir que o resultado desejado seja alcançado. Um script de normalização mal escrito pode modificar os dados, impactando diretamente o trabalho do analista.