Laboratório - Leitura de logs do servidor

Objetivos

Parte 1: Leitura de Arquivos de Log com Cat, Mais, Menos e Tail

Parte 2: Arquivos de log e Syslog

Parte 3: Arquivos de log e Jornalctl

Histórico/Cenário

Os arquivos de log são uma ferramenta importante para solução de problemas e monitoramento. Aplicação diferente gera diferentes arquivos de log, cada um contendo seu próprio conjunto de campos e informações. Enquanto a estrutura de campo pode mudar entre arquivos de log, as ferramentas usadas para lê-los são principalmente as mesmas. Neste laboratório, você aprenderá sobre ferramentas comuns usadas para ler arquivos de log e praticar como usá-las.

Recursos necessários

- Máquina Virtual CyberOps Workstation

Instruções

Parte 1: Lendo arquivos de log com Cat, More, Less e Tail

Arquivos de log são arquivos usados para registrar eventos específicos acionados por aplicativos, serviços ou o próprio sistema operacional. Normalmente armazenados como texto simples, os arquivos de log são um recurso indispensável para a solução de problemas.

Etapa 1: Abrindo Arquivos de Log.

Os arquivos de log geralmente contêm informações de texto simples que podem ser visualizadas por praticamente qualquer programa capaz de lidar com texto (editores de texto, por exemplo). No entanto, devido à conveniência, usabilidade e velocidade, algumas ferramentas são mais comumente usadas do que outras. Esta seção se concentra em quatro programas baseados em linha de comando: **cat**, **more**, **less**, e **tail**.

cat, derivado da palavra 'concatenar', é uma ferramenta UNIX baseada em linha de comando usada para ler e exibir o conteúdo de um arquivo na tela. Devido à sua simplicidade e pode abrir um arquivo de texto e exibi-lo em um terminal somente texto, **cat** é amplamente utilizado até hoje.

a. Inicie a **VM Worstation CyberOps** e abra uma janela de terminal.

b. Na janela do terminal, execute o comando abaixo para exibir o conteúdo do arquivo **logstash-tutorial.log**, localizado na pasta **/home/analyst/lab.support.files/**:

```
analyst@secOps ~$ cat /home/analyst/lab.support.files/logstash-tutorial.log
```

O conteúdo do arquivo deve rolar pela janela do terminal até que todo o conteúdo seja exibido.

Qual é a desvantagem de usar **cat** com arquivos de texto grandes?

Outra ferramenta popular para visualizar arquivos de log é **more**. Igual a **cat**, **more** também é uma ferramenta baseada em linha de comando UNIX que pode abrir um arquivo baseado em texto e exibir o conteúdo do arquivo na tela. A principal diferença entre **cat** e **more** é que **more** suporta quebras de página, permitindo que o usuário visualize o conteúdo de um arquivo, uma página por vez. Isso pode ser feito usando a barra de espaço para exibir a próxima página.

c. Na mesma janela de terminal, use o comando abaixo para exibir o conteúdo do arquivo **logstash-tutorial.log** novamente. Desta vez usando **more**:

```
analyst@secOps ~$ more /home/analyst/lab.support.files/logstash-tutorial.log
```

O conteúdo do arquivo deve rolar pela janela do terminal e parar quando uma página for exibida. Pressione a barra de espaço para avançar para a próxima página. Pressione enter para exibir a próxima linha de texto.

Qual é a desvantagem de usar **more**?

Com base na funcionalidade de **cat** e **more**, a ferramenta **less** permite que o conteúdo de um arquivo seja exibido página por página, ao mesmo tempo que permite ao usuário a opção de visualizar as páginas exibidas anteriormente.

d. Na mesma janela de terminal, use **less** para exibir o conteúdo do arquivo **logstash-tutorial.log** novamente:

```
analyst@secOps ~$ less /home/analyst/lab.support.files/logstash-tutorial.log
```

O conteúdo do arquivo deve rolar pela janela do terminal e parar quando uma página for exibida. Pressione a barra de espaço para avançar para a próxima página. Pressione enter para exibir a próxima linha de texto. Use as teclas de seta para cima e para baixo para mover para frente e para trás no arquivo de texto.

Use a tecla "**q**" no teclado para sair da ferramenta **less**.

e. O comando **tail** exibe o fim de um arquivo de texto. Por padrão, o **tail** exibe as últimas dez linhas do arquivo.

Use tail para exibir as últimas dez linhas do arquivo **/home/analyst/lab.support.files/logstash-tutorial.log**.

```
analyst@secOps ~$ tail /home/analyst/lab.support.files/logstash-tutorial.log
218.30.103.62 - - [04/Jan/2015:05:28:43 +0000] "GET /blog/geekery/xvfb-firefox.html HTTP/1.1" 200 10975 "-" "Sogou web spider/4.0(+http://www.sogou.com/docs/help/webmasters.htm#07)"
218.30.103.62 - - [04/Jan/2015:05:29:06 +0000] "GET /blog/geekery/puppet-facts-into-mcollective.html HTTP/1.1" 200 9872 "-" "Sogou web spider/4.0(+http://www.sogou.com/docs/help/webmasters.htm#07)"
198.46.149.143 - - [04/Jan/2015:05:29:13 +0000] "GET /blog/geekery/disabling-battery-in-ubuntu-vms.html?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+semicomplete%2Fmain+%28semicomplete.com+-+Jordan+Sissel%29 HTTP/1.1" 200 9316 "-" "Tiny Tiny RSS/1.11 (http://tt-rss.org/)"
198.46.149.143 - - [04/Jan/2015:05:29:13 +0000] "GET /blog/geekery/solving-good-or-bad-problems.html?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+semicomplete%2Fmain+%28semicomplete.com+-+Jordan+Sissel%29 HTTP/1.1" 200 10756 "-" "Tiny Tiny RSS/1.11 (http://tt-rss.org/)"
```

```
218.30.103.62 - - [04/Jan/2015:05:29:26 +0000] "GET /blog/geekery/jquery-interface-
puffer.html%20target= HTTP/1.1" 200 202 "-" "Sogou web
spider/4.0(+http://www.sogou.com/docs/help/webmasters.htm#07)"
218.30.103.62 - - [04/Jan/2015:05:29:48 +0000] "GET /blog/geekery/ec2-reserved-vs-
ondemand.html HTTP/1.1" 200 11834 "-" "Sogou web
spider/4.0(+http://www.sogou.com/docs/help/webmasters.htm#07)"
66.249.73.135 - - [04/Jan/2015:05:30:06 +0000] "GET /blog/web/firefox-scrolling-
fix.html HTTP/1.1" 200 8956 "-" "Mozilla/5.0 (iPhone; CPU iPhone OS 6_0 like Mac OS X)
AppleWebKit/536.26 (KHTML, like Gecko) Version/6.0 Mobile/10A5376e Safari/8536.25
(compatible; Googlebot/2.1; +http://www.google.com/bot.html)"
86.1.76.62 - - [04/Jan/2015:05:30:37 +0000] "GET /projects/xdotool/ HTTP/1.1" 200
12292 "http://www.haskell.org/haskellwiki/Xmonad/Frequently_asked_questions"
"Mozilla/5.0 (X11; Linux x86_64; rv:24.0) Gecko/20140205 Firefox/24.0
Iceweasel/24.3.0"
86.1.76.62 - - [04/Jan/2015:05:30:37 +0000] "GET /reset.css HTTP/1.1" 200 1015
"http://www.semicomplete.com/projects/xdotool/" "Mozilla/5.0 (X11; Linux x86_64;
rv:24.0) Gecko/20140205 Firefox/24.0 Iceweasel/24.3.0"
86.1.76.62 - - [04/Jan/2015:05:30:37 +0000] "GET /style2.css HTTP/1.1" 200 4877
"http://www.semicomplete.com/projects/xdotool/" "Mozilla/5.0 (X11; Linux x86_64;
rv:24.0) Gecko/20140205 Firefox/24.0 Iceweasel/24.3.0"
```

Etapa 2: Logs de Acompanhamento Ativo.

Em algumas situações, é desejável monitorar os arquivos de log à medida que as entradas de log são gravadas nos arquivos de log. Para esses casos, o comando **tail -f** é muito útil.

a. Use tail -f para monitorar ativamente o conteúdo do arquivo **/var/log/syslog** :

```
analyst@secOps ~$ sudo tail –f /home/analyst/lab.support.files/logstash-
tutorial.log
```

O que esta diferente na saída de **tail** e **tail -f**? Explique.

b. Para observar o tail –f em ação, abra uma segunda janela de terminal. Organize sua exibição para que você possa ver as duas janelas do terminal. Redimensione as janelas para que você possa vê-las ao mesmo tempo, conforme mostrado na imagem abaixo:

A janela do terminal no topo está executando o **tail -f** para monitorar o arquivo**/home/analyst/lab.support.files/logstash-tutorial.log**. Use a janela do terminal na parte inferior para adicionar informações ao arquivo monitorado.

Para facilitar a visualização, selecione a janela superior do terminal (aquela que executa **tail -f**) e pressione Enter algumas vezes. Isso adicionará algumas linhas entre o conteúdo atual do arquivo e as novas informações a serem adicionadas.

c. Selecione a janela inferior do terminal e digite o seguinte comando:

```
[analyst@secOps ~]$ echo "this is a new entry to the monitored log file" >>
lab.support.files/logstash-tutorial.log
```

O comando acima anexa a mensagem "this is a new entry to the monitored log file" ao arquivo**/home/analyst/lab.support.files/logstash-tutorial.log** . Porque tail –f está monitorando o arquivo no momento em que uma linha é adicionada a ele. A janela superior deve exibir a nova linha em tempo real.

d. Pressione CTRL + C para interromper a execução de **tail -f** e retornar ao prompt do shell.

e. Feche uma das duas janelas do terminal.

Parte 2: Arquivos de log e Syslog

Devido à sua importância, é uma prática comum concentrar os arquivos de log em um computador de monitoramento. **Syslog** é um sistema projetado para permitir que dispositivos enviem seus arquivos de log para um servidor centralizado, conhecido como servidor de **syslog**. Os clientes se comunicam com um servidor syslog usando o protocolo**syslog**. **Syslog** é comumente implantado e oferece suporte a praticamente todas as plataformas de computador.

O CyberOps Workstation VM gera arquivos de log no nível do sistema operacional e os entrega para **syslog**.

a. Use o comando **cat** como **root** para listar o conteúdo do arquivo **/var/log/syslog.1** . Este arquivo contém as entradas de log que são geradas pelo sistema operacional CyberOps Workstation VM e enviadas para o serviço**syslog**.

```
analyst@secOps ~$ sudo cat /var/log/syslog.1
[sudo] password for analyst:
Feb 7 13:23:15 secOps kernel: [ 5.458959] psmouse serio1: hgpk: ID: 10 00 64
Feb 7 13:23:15 secOps kernel: [ 5.467285] input: ImExPS/2 BYD TouchPad as
/devices/platform/i8042/serio1/input/input6
Feb 7 13:23:15 secOps kernel: [ 5.502469] RAPL PMU: API unit is 2^-32 Joules, 4 fixed
counters, 10737418240 ms ovfl timer
Feb 7 13:23:15 secOps kernel: [ 5.502476] RAPL PMU: hw unit of domain pp0-core 2^-0
Joules
Feb 7 13:23:15 secOps kernel: [ 5.502478] RAPL PMU: hw unit of domain package 2^-0
Joules
Feb 7 13:23:15 secOps kernel: [ 5.502479] RAPL PMU: hw unit of domain dram 2^-0 Joules
Feb 7 13:23:15 secOps kernel: [ 5.502480] RAPL PMU: hw unit of domain pp1-gpu 2^-0
Joules
Feb 7 13:23:15 secOps kernel: [ 5.672547] ppdev: user-space parallel port driver
Feb 7 13:23:15 secOps kernel: [ 5.709000] pcnet32 0000:00:03.0 enp0s3: renamed from
eth0
Feb 7 13:23:16 secOps kernel: [ 6.166738] pcnet32 0000:00:03.0 enp0s3: link up,
100Mbps, full-duplex
Feb 7 13:23:16 secOps kernel: [ 6.706058] random: crng init done
Feb 7 13:23:18 secOps kernel: [ 8.318984] floppy0: no floppy controllers found
Feb 7 13:23:18 secOps kernel: [ 8.319028] work still pending
Feb 7 14:26:35 secOps kernel: [ 3806.118242] hrtimer: interrupt took 4085149 ns
Feb 7 15:02:13 secOps kernel: [ 5943.582952] pcnet32 0000:00:03.0 enp0s3: link down
Feb 7 15:02:19 secOps kernel: [ 5949.556153] pcnet32 0000:00:03.0 enp0s3: link up,
100Mbps, full-duplex
```

Por que o comando **cat** teve que ser executado como **root**?

b. Observe que o arquivo **/var/log/syslog** armazena somente as entradas de log mais recentes. Para manter o arquivo syslog pequeno, o sistema operacional rotaciona periodicamente os arquivos de log, renomeando arquivos de log mais antigos como **syslog.1**, **syslog.2**e assim por diante.

Use o comando **cat** para listar arquivos **syslog** mais antigos:

```
analyst@secOps ~$ sudo cat /var/log/syslog.2
analyst@secOps ~$ sudo cat /var/log/syslog.3
analyst@secOps ~$ sudo cat /var/log/syslog.4
```

Você consegue pensar em um motivo pelo qual é tão importante manter a data e a hora dos computadores corretamente sincronizadas?

Parte 3: Arquivos de log e Journalctl

Outro sistema de gerenciamento de log popular é conhecido como **journal**. Gerenciado pelo daemon **journald**, o sistema foi projetado para centralizar o gerenciamento de logs, independentemente de onde as mensagens são originadas. No contexto deste laboratório, o recurso mais evidente do daemon do sistema **journal** é o uso de arquivos binários apenas anexados servindo como seu **arquivo de log**.

Etapa 1: Executando o journalctl sem opções.

a. Para olhar os registros do **journald**, use o comando **journalctl**. A ferramenta **journalctl** interpreta e exibe as entradas de log armazenadas anteriormente nos arquivos de log binários do **journal**.

```
analyst@secOps ~$ journalctl
-- Logs begin at Fri 2014-09-26 14:13:12 EDT, end at Tue 2017-02-07 13:23:29 ES
Sep 26 14:13:12 dataAnalyzer systemd[1087]: Starting Paths.
Sep 26 14:13:12 dataAnalyzer systemd[1087]: Reached target Paths.
Sep 26 14:13:12 dataAnalyzer systemd[1087]: Starting Timers.
Sep 26 14:13:12 dataAnalyzer systemd[1087]: Reached target Timers.
Sep 26 14:13:12 dataAnalyzer systemd[1087]: Starting Sockets.
Sep 26 14:13:12 dataAnalyzer systemd[1087]: Reached target Sockets.
Sep 26 14:13:12 dataAnalyzer systemd[1087]: Starting Basic System.
Sep 26 14:13:12 dataAnalyzer systemd[1087]: Reached target Basic System.
Sep 26 14:13:12 dataAnalyzer systemd[1087]: Starting Default.
Sep 26 14:13:12 dataAnalyzer systemd[1087]: Reached target Default.
Sep 26 14:13:12 dataAnalyzer systemd[1087]: Startup finished in 18ms.
Sep 26 14:14:24 dataAnalyzer systemd[1087]: Stopping Default.
Sep 26 14:14:24 dataAnalyzer systemd[1087]: Stopped target Default.
Sep 26 14:14:24 dataAnalyzer systemd[1087]: Stopping Basic System.
Sep 26 14:14:24 dataAnalyzer systemd[1087]: Stopped target Basic System.
Sep 26 14:14:24 dataAnalyzer systemd[1087]: Stopping Paths.
Sep 26 14:14:24 dataAnalyzer systemd[1087]: Stopped target Paths.
Sep 26 14:14:24 dataAnalyzer systemd[1087]: Stopping Timers.
Sep 26 14:14:24 dataAnalyzer systemd[1087]: Stopped target Timers.
Sep 26 14:14:24 dataAnalyzer systemd[1087]: Stopping Sockets.
<output omitted>
```

Observação: Executar o journalctl como root exibirá informações mais detalhadas.

b. Use CTRL+C para sair da tela.

Etapa 2: Journalctl e algumas opções.

Parte do poder de usar **journalctl** reside em suas opções. Para os comandos a seguir, use CRTL+C para sair da tela.

a. Use **journalctl —utc** para exibir todos os carimbos de data/hora no horário UTC:

```
analyst@secOps ~$ sudo journalctl --utc
```

b. Use **journalctl -b** para exibir entradas de log registradas durante a última inicialização:

```
analyst@secOps ~$ sudo journalctl -b
Feb 07 08:23:13 secOps systemd-journald[172]: Time spent on flushing to /var is
Feb 07 08:23:13 secOps kernel: Linux version 4.8.12-2-ARCH (builduser@andyrtr)
Feb 07 08:23:13 secOps kernel: x86/fpu: Supporting XSAVE feature 0x001: 'x87 fl
Feb 07 08:23:13 secOps kernel: x86/fpu: Supporting XSAVE feature 0x002: 'SSE re
Feb 07 08:23:13 secOps kernel: x86/fpu: Supporting XSAVE feature 0x004: 'AVX re
Feb 07 08:23:13 secOps kernel: x86/fpu: xstate_offset[2]: 576, xstate_sizes[2]
Feb 07 08:23:13 secOps kernel: x86/fpu: Enabled xstate features 0x7, context si
Feb 07 08:23:13 secOps kernel: x86/fpu: Using 'eager' FPU context switches.
Feb 07 08:23:13 secOps kernel: e820: BIOS-provided physical RAM map:
<output omitted>
```

c. Use **journalctl** para especificar o serviço e o período de tempo para entradas de log. O comando abaixo mostra todos os logs de serviço **nginx** registrados hoje:

```
analyst@secOps ~$ sudo journalctl -u nginx.service --since today
```

d. Use a opção **-k** para exibir apenas as mensagens geradas pelo kernel:

```
analyst@secOps ~$ sudo journalctl -k
```

e. Semelhante ao **tail -f** descrito acima, use a opção **-f** para seguir ativamente os logs conforme eles estão sendo escritos:

```
analyst@secOps ~$ sudo journalctl -f
```

Perguntas para reflexão

Compare Syslog e Journald. Quais são as vantagens e desvantagens de cada um?