Instituto de Matemática e Estatística da USP

MAC0216 - Técnicas de Programação I - 2s2022

EP2
Entrega até 8:00 de 17/10/2022
(INDIVIDUAL)

Prof. Daniel Macêdo Batista

1 Problema

A detecção de intrusões em sistemas computacionais é uma tarefa que precisa ser realizada continuamente, já que intrusões podem ocorrer a qualquer momento sem aviso prévio. Como não há garantia de que a pessoa responsável pela administração dos sistemas estará na frente do computador na hora dos problemas, é muito comum que sistemas de detecção de intrusão registrem alertas sempre que algo diferente do normal aconteça. Cada alerta precisa ser verificado a fim de se concluir se é um falso positivo ou não. Caso não seja um falso positivo, ações para mitigação da intrusão devem ser tomadas.

Para que os alertas sejam visualizados o mais rápido possível pela pessoa responsável pela administração dos sistemas, em um passado não muito distante eram enviadas mensagens para *pagers* ou mensagens SMS para telefones celulares. Nos últimos anos essas opções têm sido substituídas pelo uso de aplicativos de mensagem como o Telegram em *smartphones*.

Este EP vai explorar o uso de diversos comandos do bash, e utilitários do Linux acessíveis pelo bash, para que um bot envie alertas via Telegram sempre que forem detectados comportamentos fora do normal em um computador Linux.

Registros de sistemas computacionais são mais uma importante Técnica de Programação pois eles ajudam no monitoramento dos sistemas e facilitam na descoberta e correção de problemas. Um pouco mais sobre isso será visto na disciplina MAC0422 - Sistemas Operacionais.

2 Requisitos

2.1 Criação do bot no Telegram

Para que o computador possa enviar uma mensagem para você via Telegram ele precisa ter um usuário. Esse usuário vai ser de um tipo especial, chamado bot. Para criar um bot no Telegram, siga os passos descritos na página em:

`https://imasters.com.br/desenvolvimento/criando-um-bot-no-telegram-para-avisar-que-um-sistema-esta-fora-do-ar`

Leia a página até o trecho onde está escrito: “Esse é o chat id.” Ao término dessa leitura e depois de ter seguido os passos você terá duas informações importantes: o token do bot e o ID do bot. Para confirmar que tudo deu certo, rode o seguinte comando no shell do seu computador conectado à Internet:

```
# Coloque o token e o ID nos lugares informados sem o '<' e o '>' abaixo
curl -s --data "text=oi" https://api.telegram.org/bot<token aqui>/sendMessage?chat_id=<ID aqui> 1>/dev/null
```

Você deverá receber um “oi” do seu bot no Telegram. Se você receber um erro de “Comando não encontrado” no shell, significa que você precisa instalar o curl no seu computador. Em um sistema operacional baseado em Debian, o comando `sudo apt-get install curl` é suficiente para a instalação.

2.2 Servidor

O paradigma Cliente-Servidor é uma das formas de realizar uma comunicação via Internet. Nesse paradigma há um computador que fornece um serviço (Servidor) que é acessado por diversos usuários de forma remota (Clientes). Para este EP você deve iniciar um servidor no seu computador e os acessos a esse servidor serão monitorados pelo seu programa.

Usaremos o software netcat para “subir” um servidor no seu computador. Para instalar o netcat em um sistema operacional baseado em Debian basta rodar `sudo apt-get install netcat`. Para iniciar o servidor no seu computador, execute:

```
nc -k -l -p 45052
```

Para se conectar a esse servidor, abra outro terminal, no mesmo computador, e execute:

```
nc 127.0.0.1 45052
```

Mantenha os dois terminais visíveis. Digite textos seguidos de ENTER no terminal onde você executou o último comando acima. Você verá que tudo que você escrever vai aparecer no terminal onde você executou o primeiro comando. Isso confirma que de fato o seu servidor está funcionando. Pressione CTRL+c no terminal do segundo comando para encerrar a execução do cliente e CTRL+c no terminal do primeiro comando para encerrar a execução do servidor. Isso é suficiente para a continuação do EP. Caso você queira fazer algo mais realista, você pode fazer a comunicação acima a partir de outro computador na sua rede local (pode ser em sua casa, pode ser em algum laboratório do IME, pode ser em um ambiente virtual caso você crie uma máquina virtual no VirtualBox, pode ser entre o seu computador e o seu *smartphone* caso você habilite o ponto de acesso neste último, conecte o seu computador nele e instale o netcat no seu *smartphone*). Para isso você precisará saber o endereço IP do computador onde você rodou o servidor. Isso pode ser feito com o comando:

```
ip address | grep "inet " | grep -v 127.0.0.1
```

O endereço de interesse deve ser um número no formato `numero.numero.numero.numero` que vai aparecer imediatamente depois da string `inet` e imediatamente antes de uma `/`. Use esse endereço no cliente do netcat para fazer a comunicação no lugar do 127.0.0.1. Pode ser que isso não funcione caso o seu sistema tenha um *firewall* ativo. Bom, falar sobre isso e falar sobre fazer esse servidor funcionar entre redes diferentes (por exemplo, para permitir que alguém de fora da sua casa consiga acessar o seu servidor no seu computador em casa) foge do escopo desta disciplina (É assunto da disciplina MAC0352 - Redes de Computadores e Sistemas Distribuídos).

2.3 Código em bash

Um programa deverá ser escrito em bash. Os seguintes requisitos deverão ser cumpridos:

- Cinco parâmetros devem ser passados para o programa na linha de comando nesta ordem:
  - Intervalo de tempo em segundos que o programa verificará o sistema (O programa ficará em um loop repetindo as verificações a cada intervalo)
  - Caminho para um arquivo onde serão salvas as verificações
  - Limite de porcentagem de uso da CPU
  - Token do bot
  - ID do bot
- A cada intervalo de tempo, o programa deve verificar quatro informações que serão impressas na saída padrão e enviadas via bot do Telegram:
  - Data no formato `dd/mm/aaaa` e hora no formato `hh:mm:ss`
  - Quais IPs estão conectados no servidor (Se um mesmo endereço IP estiver conectado mais de uma vez, ele deve aparecer listado uma única vez). Para saber quais IPs estão conectados, use o comando `netstat -anp 2>/dev/null | grep 45052` e verifique as linhas com status `ESTABELECIDA` (pode ser que o termo seja outro caso o seu sistema não esteja configurado em Português). Caso não haja nenhuma conexão em um dado instante, nada precisará ser impresso na saída padrão e nem enviado para o Telegram
  - Quantos '%' a CPU está ociosa, caso esse valor seja menor do que o limite de porcentagem passado na linha de comando. Caso a porcentagem de ociosidade esteja acima do valor passado na linha de comando, nada precisará ser impresso na saída padrão e nem enviado para o Telegram
  - Informação de se o servidor está fora do ar (O `netstat` poderá ajudar nisso). Essa informação terá que ser repetida até que o servidor retorne
- A cada intervalo de tempo, o programa também deve salvar no arquivo passado na linha de comando as informações abaixo. Caso o arquivo já exista, ele não poderá ser apagado. As novas informações devem ser anexadas ao final do arquivo existente:
  - Data e hora, da mesma forma explicada acima
  - Os IPs que estão conectados, da mesma forma explicada acima
  - Informação de ociosidade da CPU mas, diferente do caso anterior, esses valores precisam ser salvos no arquivo mesmo se estiverem acima do limite passado na linha de comando
  - Informação sobre servidor fora do ar, da mesma forma explicada acima
- A cada 100 $\times$ intervalo de tempo, o programa também deve salvar no arquivo passado na linha de comando a informação abaixo:
  - A média das 100 últimas medições de ociosidade de CPU
  - A quantidade de tempo, em segundos, que o servidor ficou fora do ar dentro desses últimos 100 intervalos de tempo
- O programa só pode usar comandos do bash e invocar utilitários do Linux vistos em sala de aula

- Caso você utilize arquivos temporários, todos deverão ser criados em tempo de execução embaixo do `/tmp`. Sempre que o programa for executado de novo, arquivos temporários da execução anterior devem ser removidos
- Todas as recomendações sobre comentários e nomes (variáveis, funções, etc...) feitas no EP1 continuam valendo neste EP

Para verificar se a medição de ociosidade da CPU está funcionando corretamente você pode rodar o programa em C do EP1 no modo '0' e com um número bem grande na entrada padrão para que a CPU fique pouco ociosa.

Não há um formato padrão para o envio das informações via bot do telegram ou para a escrita no arquivo. O importante é que tudo que foi solicitado seja informado de uma forma que seja possível interpretar cada informação (Só apresentar os números sem informação adicional não é uma boa prática).

2.4 Relatório

O desempenho, em termos de atraso entre o momento de uma conexão ao servidor e o momento da chegada da mensagem no Telegram deverá ser avaliada durante 7 dias de semana em 3 horários diferentes: um de manhã, um de tarde e um de noite. Tente repetir as medições sempre nos mesmos horários para que a comparação seja justa.

Para medir o atraso você terá que usar um cronômetro (um relógio de verdade, um software no seu computador ou uma app no *smartphone*). Rode o seu programa com o parâmetro de intervalo em segundos igual a `1`. Inicie o cronômetro junto com a execução do netcat cliente e pare quando a mensagem chegar no Telegram. Coloque os valores numa tabela no seu relatório conforme apresentado na Tabela 1.

Tabela 1: Atraso em segundos

Dia da semana	**Data e hora**	**Atraso**
Domingo	dd/mm hh:mm manhã	Valor em segundos
Domingo	dd/mm hh:mm tarde	Valor em segundos
Domingo	dd/mm hh:mm noite	Valor em segundos
Segunda	dd/mm hh:mm manhã	Valor em segundos
Segunda	dd/mm hh:mm tarde	Valor em segundos
Segunda	dd/mm hh:mm noite	Valor em segundos
Terça	dd/mm hh:mm manhã	Valor em segundos
Terça	dd/mm hh:mm tarde	Valor em segundos
Terça	dd/mm hh:mm noite	Valor em segundos
Quarta	dd/mm hh:mm manhã	Valor em segundos
Quarta	dd/mm hh:mm tarde	Valor em segundos
Quarta	dd/mm hh:mm noite	Valor em segundos
Quinta	dd/mm hh:mm manhã	Valor em segundos
Quinta	dd/mm hh:mm tarde	Valor em segundos
Quinta	dd/mm hh:mm noite	Valor em segundos
Sexta	dd/mm hh:mm manhã	Valor em segundos
Sexta	dd/mm hh:mm tarde	Valor em segundos
Sexta	dd/mm hh:mm noite	Valor em segundos
Sábado	dd/mm hh:mm manhã	Valor em segundos
Sábado	dd/mm hh:mm tarde	Valor em segundos
Sábado	dd/mm hh:mm noite	Valor em segundos

Além das tabelas, o seu relatório precisa ter um cabeçalho com seu nome e NUSP e uma conclusão explicando se os resultados obtidos em termos de atraso seguem algum padrão que faça sentido considerando os diferentes dias da semana e horários. Também informe a configuração dos computadores usados (processador, memória e sistema operacional) onde todas as execuções foram realizadas. O ideal é que todas as medições sejam feitas no mesmo computador mas caso isso seja complicado para você, por conta da necessidade de execuções em horários específicos, tudo bem se forem usados computadores diferentes.

2.5 LEIAME

Junto com os programas e com o relatório você terá que entregar também um arquivo LEIAME em texto puro (Ele pode ser nomeado apenas como LEIAME ou pode ser LEIAME.txt). Crie o seu arquivo com no mínimo estas cinco seções dentro dele:

```
AUTOR:
<Seu nome, NUSP e endereço de e-mail>

DESCRIÇÃO:
<Explique o que o programa faz (Não diga apenas que ele é o EP2 da
disciplina tal, explique o que de fato o programa faz)>

COMO EXECUTAR:
<Informe como rodar o programa, como passar parâmetros
para ele e como interpretar a saída>

TESTES:
<Dê exemplos de execuções que sejam suficientes para que alguém
confirme que o programa está funcionando corretamente>

DEPENDÊNCIAS:
<Informe o que é necessário para rodar o programa. Informações como:
versão do bash, informações do SO onde você executou e sabe que ele
funciona são importantes de serem colocadas aqui.>
```

3 Entrega

Você deverá entregar um arquivo tarball comprimido (.tar.gz) contendo os seguintes itens:

- 1 único arquivo .sh com o código fonte do programa em bash;
- 1 arquivo LEIAME (pode ser LEIAME.txt) em texto puro;
- 1 arquivo .pdf com o relatório.

Todos esses arquivos devem estar presentes. Caso algum arquivo esteja faltando, o EP não será corrigido e a nota dele será zero.

O desempacotamento do tarball deve produzir um diretório contendo os itens. O nome do diretório deve ser ep2-seu_nome. Por exemplo: ep2-joao_dos_santos.

A entrega do tarball deve ser feita no e-Disciplinas.

O EP deve ser feito individualmente.

Obs.1: Serão descontados 2,0 pontos de EPs com tarballs que não estejam nomeados como solicitado ou que não criem o diretório com o nome correto após serem descompactados. Confirme que o seu tarball está correto, descompactando ele no shell (não confie em interfaces gráficas na hora de verificar o seu tarball pois alguns gerenciadores de arquivos criam o diretório automaticamente mesmo quando esse diretório não existe).

Obs.2: A depender da qualidade do conteúdo que for entregue, o EP pode ser considerado como não entregue, implicando em MF=0,0. Isso acontecerá também se for enviado um tarball corrompido, códigos fonte vazios ou códigos fonte que resolvam um problema completamente diferente do que foi solicitado.

Obs.3: O prazo de entrega expira às 8:00:00 do dia 17/10/2022.

4 Avaliação

80% da nota será dada pela implementação, 10% pelo LEIAME e 10% pelo relatório. Os critérios detalhados da correção serão disponibilizados apenas quando as notas forem liberadas.