HACKERS DO BEM

Desenvolvimento de Scripts

Scripts em Linux

Aula 01 – Introdução ao desenvolvimento de scripts em Linux

- **O que é Script?**
    - Série de instruções
    - Otimizar processos

- **Por que usar scripts?**

- **Automatização de processos.**

- **Importância da automação de tarefas:**

  - Eficiência;
  - Consistência;
  - Resposta rápida a incidentes;
  - Análise de dados;
  - Monitoramento contínuo;
  - Escalonamento

- Noções da Linguagem Bash:

Comentários: Linhas começando com # são comentários e são ignorados pelo interpretador.

Variáveis: Variáveis são usadas para armazenar valores. Exemplo: nome=“Usuário”

Comandos: Comandos do sistema podem ser executados usando crases (``) ou $(). Exemplo: data= $(date)

Estruturas Condicionais: if, elif e fi são usados para tomada de decisões baseadas em condições.

Estruturas de Loop: for e while permitem repetir ações várias vezes.

Redirecionamento de Saída: > (redireciona saída para um arquivo) e >> (anexa saída a um arquivo).

Testes e Comparação de Strings: O comando test (ou[ ]) é usado para comparar strings, números e arquivos.

Funções: Permitem definir blocos de código reutilizáveis.

Argumentos da Linha de Comando: Variáveis especiais como $1, $ 2

Interpolação de Variáveis: Variáveis podem ser usadas dentro de strings com aspas duplas para interpolação.

Aula 02 – Exemplos práticos:

- scripts básicos para automação de tarefas de segurança cibernética em Linux;

- Um script que analisa os logs de autenticação para identificar tentativas de login falhadas pode ajudar a detectar possíveis tentativas de invasão. O comando `grep` pode ser usado para procurar padrões de logs.

Exemplo de verificação de Logs de Autenticação:

```
#!/bin/bash
echo "Procurando por tentativas de login falhadas nos logs:"
grep "Failed password" /var/log/auth.log
```

- Automatizar a verificação e instalação de atualizações do sistema é importante para manter o sistema seguro. O uso de comandos como `apt-get`(para sistemas baseados em Debian) ou `yum` (para sistemas baseados em Red Hat) pode ser incorporado em um script.

Exemplo de Verificação de Atualizações do Sistema:

```
#!/bin/bash

echo “Verificando atualizações disponíveis...”

apt-get update

apt-get upgrade -s | grep “upgraded”
```

- Usando ferramentas como o `curl`, você pode criar um script que verifica automaticamente a presença de vulnerabilidades conhecidas em sites.

Exemplo de Verificação de Vulnerabilidades em Sites:

```
#!/bin/bash
site="https://www.google.com"
vulnerabilidade="CVE-2014-6271"
echo
resultado=$(curl -s -I "$site" | grep "$vulnerabilidade")
if [ -n "$resultado" ]; then
    echo "Vulnerabilidade Shellshock detectada no site $site."
else
    echo "Site seguro, nenhuma vulnerabilidade Shellshock detectada."
fi
```

Aula 03 – Introdução ao desenvolvimento de scripts em Linux

Substituição de Comando com `$()`:

- Nessa abordagem, você coloca o comando entre parênteses e precede-o com o símbolo de dólar seguido de parênteses. O resultado do comando é armazenado em uma variável.

Por exemplo:

````bash
resultado=$(ls /caminho/do/diretorio)
echo "Conteúdo do diretório: $resultado"
```

## Substituição de Comando com Backticks (``):

- Essa é uma abordagem mais antiga, onde você coloca o comando entre crases (backticks) e o resultado é novamente armazenado em uma variável.

Por exemplo:

```bash
resultado=`ls /caminho/do/diretorio`
echo "Conteúdo do diretório: $resultado"
```

Geralmente, a primeira abordagem com `$()` é preferida por ser mais legível e mais fácil de ser aninhada.

## Exemplo de Execução de Comando Externo em Script:

Suponha que você queira verificar o espaço livre em disco usando o comando `df -h`. Você pode incorporar esse comando em um script da seguinte maneira:

```bash
#!/bin/bash

espaco_livre=$(df -h)

echo "Espaço livre em disco:"

echo "$espaco_livre"
```

Nesse exemplo, o comando `df -h` é executado e seu resultado é armazenado na variável `espaco_livre`. Em seguida, o resultado é exibido usando o comando `echo`.

# Exemplos Práticos de Scripts para Operações de Arquivo, Diretório e Execução de Comandos:

1. Criando um Diretório e Copiando um Arquivo:

```bash
#!/bin/bash
echo "Criando um diretório e copiando um arquivo..."
mkdir novo_diretorio
cp arquivo.txt novo_diretorio/
echo "Diretório criado e arquivo copiado."
```

2. Listando Conteúdo de um Diretório e Renomeando um Arquivo:

```bash
#!/bin/bash
echo "Listando conteúdo do diretório e renomeando um
arquivo..."
ls /caminho/do/diretorio
mv antigo_nome.txt novo_nome.txt
echo "Arquivo renomeado."
```

3. Removendo um Arquivo e Executando um Comando Externo:

```bash
#!/bin/bash
echo "Removendo um arquivo e executando um comando externo..."
rm arquivo_removido.txt
resultado=$(ls /caminho/do/diretorio)
echo "Conteúdo do diretório: $resultado"
```

# Aula 04 – Introdução ao desenvolvimento de scripts em Linux

## Script de Detecção de Portas Abertas:

1. Um script que verifica portas abertas em um sistema e alerta sobre portas não autorizadas ou suspeitas.

```
#!/bin/bash
host="127.0.0.1"
portas=("80" "22" "443" "3389")
for porta in "${portas[@]}"; do
    nc -zv "$host" "$porta" > /dev/null 2>&1
    if [ $? -eq 0 ]; then
        echo "Porta $porta está aberta em $host"
    else
        echo "Porta $porta está fechada em $host"
    fi
done
```

## Script de Análise de Logs de Autenticação:

2. Um script que analisa logs de autenticação em busca de padrões suspeitos, como múltiplas tentativas de login falhadas.

```
#!/bin/bash
log_file="/var/log/auth.log"
padrao="Failed password"
if grep -q "$padrao" "$log_file"; then
    echo "Padrão suspeito encontrado nos logs de autenticação."
else
    echo "Nenhum padrão suspeito encontrado nos logs de autenticação."
fi
```

3. Script de Detecção de Ataques de Força Bruta:
- Um script que monitora logs em busca de atividades de força bruta em tentativas de acesso, como tentativas repetidas de login.

```
#!/bin/bash
log_file="/var/log/auth.log"
limite_tentativas=5
tentativas=$(grep "Failed password" "$log_file" | wc -l)
if [ "$tentativas" -ge "$limite_tentativas" ]; then
    echo "Possível ataque de força bruta detectado."
else
    echo "Nenhuma atividade de força bruta detectada."
fi
```

## 4. Script de Verificação de Vulnerabilidades em Sistemas:

Um script que verifica sistemas em busca de vulnerabilidades conhecidas, usando bancos de dados de CVEs (Vulnerabilidades e Exposições Comuns).

```
#!/bin/bash
sistema="Ubuntu"
versao="20.04"
vulnerabilidade=$(grep "$sistema $versao" cve_database.txt)
if [ -n "$vulnerabilidade" ]; then
    echo "Vulnerabilidade conhecida encontrada no sistema
$sistema $versao."
else
    echo "Nenhuma vulnerabilidade conhecida encontrada."
fi
```

5. Script de Monitoramento de Arquivos Críticos:
Um script que monitora alterações em arquivos críticos do sistema e alerta sobre quaisquer modificações não autorizadas.

```
#!/bin/bash

diretorio="/diretorio_critico"
while true; do
    changes=$(inotifywait -e modify,create,delete "$diretorio")
    echo "Alterações detectadas em: $changes"
done
```

## 6. Script de Análise Forense para Coleta de Evidências:

Um script que automatiza a coleta de evidências de um sistema comprometido para análise forense posterior.

```
#!/bin/bash
output_dir="/evidencias"
log_file="/var/log/syslog"

mkdir -p "$output_dir"
cp "$log_file" "$output_dir/syslog_copia.log"
echo "Evidências coletadas em $output_dir"
```


HACKERS
DO
BEM