Documentação do Projeto: Script de Diagnóstico de Rede

Documentação do Projeto com Código

Introdução

Este script é uma ferramenta prática para diagnosticar dispositivos em uma rede local. Ele verifica se os dispositivos estão online, coleta informações como latência e MAC address, realiza varreduras de portas e gera um relatório visual e interativo em formato HTML, incluindo uma tabela e um gráfico com os resultados.

Funcionamento do Script

- Verifica status online/offline dos dispositivos com ping.
- Coleta latência e endereço MAC.
- Oferece opção de escaneamento de portas com nmap (rápido ou profundo).
- Gera um relatório HTML com tabela e gráfico (Chart.js).

Explicação do Código

1. Criação do Arquivo HTML

Cria o arquivo relatorio_dispositivos.html e escreve toda a estrutura inicial da página.

Trecho do código:

```
RELATORIO="relatorio_dispositivos.html"
cat << EOF > "$RELATORIO"
<!DOCTYPE html>
<html>
...
```

2. Definição dos Dispositivos

Define uma lista com IPs e nomes dos dispositivos da rede.

Trecho do código:

```
declare -A dispositivos=(
    ["PC"]="192.168.0.2"
    ["notebook"]="192.168.0.3"
    ["Modem"]="192.168.0.1"
)
```

Documentação do Projeto: Script de Diagnóstico de Rede

3. Loop para Verificar Cada Dispositivo

Passa por cada dispositivo da lista para realizar diagnóstico.

Trecho do código:

```
for nome in "${!dispositivos[@]}"; do
    check_device "$nome" "${dispositivos[$nome]}"
done
```

4. Verificando se o Dispositivo Está Online

Usa ping para checar se está ativo.

Trecho do código:

```
ping -c 1 -W 1 "$ip" &>/dev/null
if [[ $? -eq 0 ]]; then
```

5. Se Online: Coletando Latência e MAC Address

Extrai latência com ping e MAC com comando arp.

Trecho do código:

```
latencia=$(ping -c 1 -W 1 "$ip" | grep 'time=' | awk -F'time=' '{print $2}' | awk '{print $1}')
mac=$(get_mac "$ip")
```

6. Perguntando sobre Deep Scan com Nmap

Solicita ao usuário se deseja uma varredura profunda nas portas.

Trecho do código:

```
read -p "Deseja fazer Deep Scan (todas as portas) no $nome? (s/n): " escolha
```

7. Executando Nmap

Executa varredura de portas com nmap (todas as portas ou top 10).

Trecho do código:

```
if [[ "$escolha" == "s" ]]; then
    portas=$(nmap -Pn -p- "$ip" | awk '/^[0-9]+\/tcp/ {print $1 " - " $2 " - " $3}' ...)
```

Documentação do Projeto: Script de Diagnóstico de Rede

```
else
    portas=$(nmap -Pn --top-ports 10 "$ip" | awk '/^[0-9]+\/tcp/ {print $1 " - " $2 " - " $3}' ...)
```

8. Se Dispositivo Estiver Offline

Define status como OFFLINE e outros dados como N/A.

Trecho do código:

```
else
    local status="<span class='offline'>OFFLINE</span>"
    ((offline_count++))
```

9. Atualizando a Tabela HTML

Adiciona uma nova linha com os dados coletados.

Trecho do código:

```
cat << EOF >> "$RELATORIO"
<h2>$nome</h2>
<table>
    <tr><th>IP</th><td>$ip</td></tr>
    ...
```

10. Atualizando o Gráfico

Conta total de dispositivos, online e offline para o gráfico.

Trecho do código:

```
cat << EOF >> "$RELATORIO"
<script>
    statusChart.data.datasets[0].data = [$online_count, $offline_count];
    statusChart.update();
</script>
```

11. Fechando o HTML e Criando o Gráfico

Finaliza o HTML e cria o gráfico com Chart.js.

Trecho do código:

Documentação do Projeto: Script de Diagnóstico de Rede

cat << EOF >> "$RELATORIO"
Relatório gerado automaticamente com Bash + Nmap + Chart.js
</body>
</html>

12. Abrindo o Relatório no Navegador
Usa xdg-open para abrir o relatório gerado no navegador.

Trecho do código:
if command -v xdg-open &>/dev/null; then
    xdg-open "$RELATORIO"

Resumo
O script faz diagnóstico com ping, arp e nmap e gera um relatório HTML com tabela e gráfico visual.

Diferenciais
- Relatório visual (HTML + gráfico)
- Ferramentas padrão do Linux
- Código comentado e modular

Conclusão
Agora você entende exatamente cada bloco do código, útil para estudo ou apresentação profissional.