Roteiro - Aula 03 - HTTPS e Servidores Web

Roteiro de Laboratório: HTTPS e Servidores Web

Objetivos

- Configurar um servidor web básico
- Implementar HTTPS em um servidor web
- Comparar o desempenho entre HTTP/1.1 e HTTP/2
- Analisar certificados digitais e seu funcionamento

Pré-requisitos

- Acesso a uma máquina Linux (física ou virtual)
- Privilégios de administrador (sudo)
- Conexão à internet para download de pacotes
- Ferramentas básicas: curl, openssl, ab (Apache Benchmark), navegador web

Material Necessário

- Computador com distribuição Linux instalada (Ubuntu/Debian recomendado)
- Navegador com ferramentas de desenvolvedor (Chrome ou Firefox)
- Editor de texto (vim, nano, VSCode, etc.)

Atividade 1: Configuração Básica de Servidor Web

Objetivos Específicos

- Instalar e configurar o servidor web Apache ou Nginx
- Criar uma página web simples
- Verificar logs e resolver problemas comuns

Passo 1: Instalação do Servidor Web

Opção A: Apache

```
# Atualizar lista de pacotes
sudo apt update
```

```
# Instalar Apache
sudo apt install apache2 -y

# Verificar status
sudo systemctl status apache2

# Iniciar o serviço (caso não esteja rodando)
sudo systemctl start apache2

# Configurar para iniciar automaticamente
sudo systemctl enable apache2
```

Opção B: Nginx

```
# Atualizar lista de pacotes
sudo apt update

# Instalar Nginx
sudo apt install nginx -y

# Verificar status
sudo systemctl status nginx

# Iniciar o serviço (caso não esteja rodando)
sudo systemctl start nginx

# Configurar para iniciar automaticamente
sudo systemctl enable nginx
```

Passo 2: Criação de Página Web Simples

Para Apache:

```
# Criar arquivo HTML
sudo nano /var/www/html/index.html
```

Para Nginx:

```
# Criar arquivo HTML
sudo nano /var/www/html/index.html
```

Insira o seguinte conteúdo HTML:

```
<!DOCTYPE html>
<html lang="pt-BR">
```

```html
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Teste de Servidor Web</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            line-height: 1.6;
            max-width: 800px;
            margin: 0 auto;
            padding: 20px;
        }
        h1 {
            color: #2c3e50;
            text-align: center;
        }
        .info {
            background-color: #f8f9fa;
            border-left: 4px solid #2c3e50;
            padding: 15px;
            margin: 20px 0;
        }
    </style>
</head>
<body>
    <h1>Página de Teste</h1>
    <div class="info">
        <p>Esta é uma página de teste para o laboratório de Redes de
Computadores 2.</p>
        <p>Servidor: <span id="server-info">Verificando...</span></p>
        <p>Protocolo: <span id="protocol">Verificando...</span></p>
        <p>Data e hora: <span id="datetime">Verificando...</span></p>
    </div>

    <script>
        // Detectar servidor
        fetch('/')
            .then(response => {
                document.getElementById('server-info').textContent =
response.headers.get('server') || 'Não identificado';
                document.getElementById('protocol').textContent =
window.location.protocol;
                document.getElementById('datetime').textContent = new
Date().toLocaleString();
            })
            .catch(error => {
                console.error('Erro ao detectar servidor:', error);
            });
    </script>
```

```
</body>
</html>
```

Passo 3: Verificação de Logs e Resolução de Problemas

Apache:

```
# Verificar logs de acesso
sudo tail -f /var/log/apache2/access.log

# Verificar logs de erro
sudo tail -f /var/log/apache2/error.log

# Verificar configuração
sudo apachectl configtest
```

Nginx:

```
# Verificar logs de acesso
sudo tail -f /var/log/nginx/access.log

# Verificar logs de erro
sudo tail -f /var/log/nginx/error.log

# Verificar configuração
sudo nginx -t
```

Passo 4: Reiniciar e Testar o Servidor

Apache:

```
# Reiniciar o servidor
sudo systemctl restart apache2
```

Nginx:

```
# Reiniciar o servidor
sudo systemctl restart nginx
```

- Abra o navegador e acesse: http://localhost ou http://127.0.0.1
- Verifique se a página é exibida corretamente

Atividade 2: Implementação de HTTPS

Objetivos Específicos

- Gerar certificado auto-assinado com OpenSSL
- Configurar o módulo SSL no servidor
- Implementar redirecionamento HTTP para HTTPS
- Testar a conexão segura
- Analisar o certificado no navegador

Passo 1: Instalação de Ferramentas Necessárias

```
# Instalar OpenSSL (provavelmente já instalado)
sudo apt install openssl -y

# Para Apache, habilitar o módulo SSL
sudo a2enmod ssl
sudo a2enmod headers  # Para redirecionamento
```

Passo 2: Geração de Certificado Auto-assinado

```
# Criar diretório para os certificados
sudo mkdir -p /etc/ssl/localcerts

# Gerar chave privada
sudo openssl genrsa -out /etc/ssl/localcerts/server.key 2048

# Gerar Certificate Signing Request (CSR)
sudo openssl req -new -key /etc/ssl/localcerts/server.key -out
/etc/ssl/localcerts/server.csr

# Ao preencher as informações, preste atenção no "Common Name" - use
localhost ou seu IP

# Gerar certificado auto-assinado (válido por 365 dias)
sudo openssl x509 -req -days 365 -in /etc/ssl/localcerts/server.csr -
signkey /etc/ssl/localcerts/server.key -out /etc/ssl/localcerts/server.crt

# Ajustar permissões
sudo chmod 644 /etc/ssl/localcerts/server.crt
sudo chmod 600 /etc/ssl/localcerts/server.key
```

Passo 3: Configuração do Servidor para HTTPS

Para Apache:

```
# Criar arquivo de configuração para o site HTTPS
sudo nano /etc/apache2/sites-available/default-ssl.conf
```

Adicione ou modifique para o seguinte conteúdo:

```
<IfModule mod_ssl.c>
    <VirtualHost _default_:443>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        SSLEngine on
        SSLCertificateFile /etc/ssl/localcerts/server.crt
        SSLCertificateKeyFile /etc/ssl/localcerts/server.key

        <FilesMatch "\.(cgi|shtml|phtml|php)$">
            SSLOptions +StdEnvVars
        </FilesMatch>

        <Directory /usr/lib/cgi-bin>
            SSLOptions +StdEnvVars
        </Directory>
    </VirtualHost>
</IfModule>
```

Configurar o redirecionamento HTTP para HTTPS:

```
sudo nano /etc/apache2/sites-available/000-default.conf
```

Adicione dentro da seção `<VirtualHost *:80>`:

```
RewriteEngine On
RewriteCond %{HTTPS} off
RewriteRule ^ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]
```

Habilite os sites e o módulo de reescrita:

```
sudo a2enmod rewrite
sudo a2ensite default-ssl
sudo systemctl restart apache2
```

Para Nginx:

```
# Editar o arquivo de configuração padrão
sudo nano /etc/nginx/sites-available/default
```

Substitua o conteúdo por:

```
server {
    listen 80;
    server_name localhost;

    # Redirecionamento para HTTPS
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl http2;
    server_name localhost;

    ssl_certificate /etc/ssl/localcerts/server.crt;
    ssl_certificate_key /etc/ssl/localcerts/server.key;

    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_prefer_server_ciphers on;
    ssl_ciphers "EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH";

    root /var/www/html;
    index index.html index.htm;

    location / {
        try_files $uri $uri/ =404;
    }
}
```

Reinicie o servidor:

```
sudo systemctl restart nginx
```

Passo 4: Teste de Conexão Segura

- Abra o navegador e acesse: https://localhost ou https://127.0.0.1
- Você verá um aviso de segurança (isso é normal com certificados auto-assinados)
- Clique em "Avançado" e depois em "Prosseguir para localhost (não seguro)"
- Verifique se a página carrega corretamente
- Confirme que o redirecionamento HTTP para HTTPS está funcionando acessando http://localhost

Passo 5: Análise do Certificado no Navegador

- Clique no ícone de cadeado na barra de endereços
- Selecione "Certificado" ou opção similar
- Analise as informações do certificado:
  - Emissor (deve ser o mesmo que você informou na criação)
  - Validade
  - Algoritmo de assinatura
  - Chave pública

Atividade 3: Comparação HTTP/1.1 vs HTTP/2

Objetivos Específicos

- Configurar suporte a HTTP/2 no servidor
- Criar uma página com múltiplos recursos
- Medir o tempo de carregamento em HTTP/1.1 e HTTP/2
- Analisar os resultados com DevTools

Passo 1: Configuração do HTTP/2

Para Apache:

Verifique se o módulo HTTP/2 está instalado e habilitado:

```
sudo apt install libapache2-mod-http2 -y
sudo a2enmod http2
```

Edite o arquivo de configuração SSL:

```
sudo nano /etc/apache2/sites-available/default-ssl.conf
```

Adicione a seguinte linha dentro da seção `<VirtualHost>`:

```
Protocols h2 http/1.1
```

Reinicie o Apache:

```
sudo systemctl restart apache2
```

Para Nginx:

O HTTP/2 já deve estar configurado na configuração anterior (pelo parâmetro `http2` na linha `listen 443 ssl http2;`).

Se não estiver, edite o arquivo de configuração:

```
sudo nano /etc/nginx/sites-available/default
```

E certifique-se de que a linha contenha `http2`:

```
listen 443 ssl http2;
```

Reinicie o Nginx:

```
sudo systemctl restart nginx
```

Passo 2: Criação de Página com Múltiplos Recursos

Crie uma pasta para o teste:

```
sudo mkdir -p /var/www/html/http2-test
cd /var/www/html/http2-test
```

Crie o arquivo HTML principal:

```
sudo nano index.html
```

Insira o seguinte conteúdo:

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Teste HTTP/2 vs HTTP/1.1</title>
    <link rel="stylesheet" href="styles.css">
    <script src="script.js" defer></script>
</head>
<body>
    <header>
        <h1>Comparação HTTP/1.1 vs HTTP/2</h1>
        <p id="protocol-version">Protocolo: Carregando...</p>
    </header>
```

```
    <main>
        <section class="image-gallery">
            <h2>Galeria de Imagens</h2>
            <div class="images">
                <!-- 20 imagens para simular muitos recursos -->
                <img src="img/img1.jpg" alt="Imagem 1">
                <img src="img/img2.jpg" alt="Imagem 2">
                <img src="img/img3.jpg" alt="Imagem 3">
                <img src="img/img4.jpg" alt="Imagem 4">
                <img src="img/img5.jpg" alt="Imagem 5">
                <img src="img/img6.jpg" alt="Imagem 6">
                <img src="img/img7.jpg" alt="Imagem 7">
                <img src="img/img8.jpg" alt="Imagem 8">
                <img src="img/img9.jpg" alt="Imagem 9">
                <img src="img/img10.jpg" alt="Imagem 10">
                <img src="img/img11.jpg" alt="Imagem 11">
                <img src="img/img12.jpg" alt="Imagem 12">
                <img src="img/img13.jpg" alt="Imagem 13">
                <img src="img/img14.jpg" alt="Imagem 14">
                <img src="img/img15.jpg" alt="Imagem 15">
                <img src="img/img16.jpg" alt="Imagem 16">
                <img src="img/img17.jpg" alt="Imagem 17">
                <img src="img/img18.jpg" alt="Imagem 18">
                <img src="img/img19.jpg" alt="Imagem 19">
                <img src="img/img20.jpg" alt="Imagem 20">
            </div>
        </section>

        <section class="results">
            <h2>Resultados</h2>
            <div id="performance">
                <p>Tempo de carregamento: <span id="load-
time">Aguardando...</span></p>
                <p>Recursos carregados: <span id="resources-
count">0</span></p>
                <p>Bytes transferidos: <span id="bytes-
transferred">0</span></p>
            </div>
        </section>
    </main>

    <footer>
        <p>Laboratório de Redes de Computadores 2 - IFC</p>
    </footer>
</body>
</html>
```

Crie o arquivo CSS:

```
sudo nano styles.css
```

Insira o seguinte conteúdo:

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: Arial, sans-serif;
    line-height: 1.6;
    padding: 20px;
    max-width: 1200px;
    margin: 0 auto;
}

header, footer {
    text-align: center;
    padding: 20px 0;
    background-color: #f4f4f4;
    margin-bottom: 20px;
}

footer {
    margin-top: 20px;
    margin-bottom: 0;
}

h1, h2 {
    margin-bottom: 15px;
    color: #333;
}

.images {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
    gap: 10px;
    margin-bottom: 30px;
}

.images img {
    width: 100%;
    height: 150px;
    object-fit: cover;
    border-radius: 5px;
    border: 1px solid #ddd;
```

```
}

.results {
    background-color: #f9f9f9;
    padding: 20px;
    border-radius: 5px;
    margin-bottom: 20px;
}

#performance p {
    margin-bottom: 10px;
    font-size: 16px;
}

#performance span {
    font-weight: bold;
    color: #2980b9;
}
```

Crie o arquivo JavaScript:

```
sudo nano script.js
```

Insira o seguinte conteúdo:

```
// Registrar tempo de início
const startTime = performance.now();

// Função para mostrar o protocolo
function detectProtocol() {
    // Detecção aproximada pelo navegador
    const connection = navigator.connection ||
                       navigator.mozConnection ||
                       navigator.webkitConnection;

    if (connection && connection.effectiveType) {
        document.getElementById('protocol-version').textContent +=
            ` ${connection.effectiveType} (Conexão)`;
    }

    // Verificar HTTP/2 consultando a própria página
    fetch(window.location.href)
        .then(response => {
            // Chrome expõe essa informação
            if (response.type === 'cors' || response.type === 'basic') {
                // Verificar aproximadamente
                if (window.chrome) {
                    document.getElementById('protocol-
```

```
version').textContent =
                        `Protocolo: Provavelmente HTTP/2 (Chrome)`;
                } else {
                    document.getElementById('protocol-
version').textContent =
                        `Protocolo:
${window.location.protocol.replace(':', '')}`;
                }
            }
        })
        .catch(error => {
            console.error('Erro ao detectar protocolo:', error);
        });
}

// Função para medir o tempo de carregamento
function measurePerformance() {
    const endTime = performance.now();
    const loadTime = ((endTime - startTime) / 1000).toFixed(2);

    document.getElementById('load-time').textContent = `${loadTime}s`;

    // Contar recursos carregados
    const resources = performance.getEntriesByType('resource');
    document.getElementById('resources-count').textContent =
resources.length;

    // Calcular bytes transferidos (aproximado)
    let totalBytes = 0;
    resources.forEach(resource => {
        if (resource.transferSize) {
            totalBytes += resource.transferSize;
        }
    });

    // Converter para KB
    const kbTransferred = (totalBytes / 1024).toFixed(2);
    document.getElementById('bytes-transferred').textContent =
`${kbTransferred} KB`;
}

// Executar quando a página carregar
window.addEventListener('load', () => {
    detectProtocol();
    measurePerformance();
});
```

Crie uma pasta para as imagens e baixe algumas imagens de amostra:

```
sudo mkdir -p /var/www/html/http2-test/img
cd /var/www/html/http2-test/img

# Baixe algumas imagens de amostra (você pode usar outras fontes)
sudo wget -O img1.jpg https://picsum.photos/200/300
sudo wget -O img2.jpg https://picsum.photos/201/300
sudo wget -O img3.jpg https://picsum.photos/202/300
sudo wget -O img4.jpg https://picsum.photos/203/300
sudo wget -O img5.jpg https://picsum.photos/204/300
sudo wget -O img6.jpg https://picsum.photos/205/300
sudo wget -O img7.jpg https://picsum.photos/206/300
sudo wget -O img8.jpg https://picsum.photos/207/300
sudo wget -O img9.jpg https://picsum.photos/208/300
sudo wget -O img10.jpg https://picsum.photos/209/300
sudo wget -O img11.jpg https://picsum.photos/210/300
sudo wget -O img12.jpg https://picsum.photos/211/300
sudo wget -O img13.jpg https://picsum.photos/212/300
sudo wget -O img14.jpg https://picsum.photos/213/300
sudo wget -O img15.jpg https://picsum.photos/214/300
sudo wget -O img16.jpg https://picsum.photos/215/300
sudo wget -O img17.jpg https://picsum.photos/216/300
sudo wget -O img18.jpg https://picsum.photos/217/300
sudo wget -O img19.jpg https://picsum.photos/218/300
sudo wget -O img20.jpg https://picsum.photos/219/300

# Ajustar permissões
sudo chmod -R 755 /var/www/html/http2-test
```

Passo 3: Medição de Tempo de Carregamento

Forçar HTTP/1.1

Modifique temporariamente a configuração para desabilitar HTTP/2:

Para Apache:

```
sudo nano /etc/apache2/sites-available/default-ssl.conf
```

Altere a linha:

```
Protocols h2 http/1.1
```

Para:

```
Protocols http/1.1
```

Reinicie o Apache:

```
sudo systemctl restart apache2
```

Para Nginx:

```
sudo nano /etc/nginx/sites-available/default
```

Altere a linha:

```
listen 443 ssl http2;
```

Para:

```
listen 443 ssl;
```

Reinicie o Nginx:

```
sudo systemctl restart nginx
```

Teste com HTTP/1.1:

1. Abra o navegador Chrome ou Firefox
2. Abra as ferramentas de desenvolvedor (F12)
3. Vá para a aba Network (Rede)
4. Marque a opção "Disable cache" (Desabilitar cache)
5. Acesse https://localhost/http2-test/
6. Anote o tempo de carregamento mostrado na página e no painel Network
7. Anote o número de recursos carregados
8. Verifique o protocolo utilizado no painel Network (h1 ou http/1.1)

Habilitar HTTP/2:

Reverta as alterações feitas para desabilitar o HTTP/2:

Para Apache:

```
sudo nano /etc/apache2/sites-available/default-ssl.conf
```

Altere a linha:

```
Protocols http/1.1
```

De volta para:

```
Protocols h2 http/1.1
```

Reinicie o Apache:

```
sudo systemctl restart apache2
```

Para Nginx:

```
sudo nano /etc/nginx/sites-available/default
```

Altere a linha:

```
listen 443 ssl;
```

De volta para:

```
listen 443 ssl http2;
```

Reinicie o Nginx:

```
sudo systemctl restart nginx
```

Teste com HTTP/2:

1. Repita o mesmo procedimento do teste HTTP/1.1
2. Verifique se o protocolo mostrado agora é h2
3. Compare os tempos de carregamento e número de recursos

Passo 4: Análise com Chrome DevTools

1. Com as ferramentas de desenvolvedor abertas, analise:
   - A visualização em cascata das requisições
   - A linha Protocol na aba Network para confirmar o protocolo
   - O tempo total de carregamento (DOMContentLoaded e Load)
   - O número de conexões estabelecidas (linhas horizontais na visualização em cascata)
2. Crie uma tabela comparativa com os resultados:

Métrica	HTTP/1.1	HTTP/2	Melhoria (%)
Tempo total de carregamento			
DOMContentLoaded			
Número de conexões			
Tamanho transferido			

1. Analise as razões para as diferenças observadas:
   - Multiplexação de streams no HTTP/2
   - Head-of-line blocking no HTTP/1.1
   - Compressão de cabeçalhos no HTTP/2

Perguntas para Reflexão

1. Qual a importância do HTTPS para a segurança na web?
2. Quais são as principais diferenças entre HTTP/1.1 e HTTP/2?
3. Por que certificados auto-assinados não são recomendados para ambientes de produção?
4. Em quais situações o HTTP/2 apresenta maior vantagem sobre o HTTP/1.1?
5. Quais são as principais diferenças entre os servidores Apache e Nginx?

Entrega do Relatório

Elabore um relatório contendo:

1. Nome dos integrantes do grupo
2. Capturas de tela das configurações realizadas
3. Resultados dos testes comparativos
4. Respostas às perguntas de reflexão
5. Conclusões sobre a atividade prática

Referências

- Documentação oficial do Apache: https://httpd.apache.org/docs/
- Documentação oficial do Nginx: https://nginx.org/en/docs/
- RFC 8446 - TLS 1.3: https://tools.ietf.org/html/rfc8446
- RFC 7540 - HTTP/2: https://tools.ietf.org/html/rfc7540