UNIPAC
Barbacena

Centro Universitário Presidente Antônio Carlos
Programação para Internet

Upload e Download

Felipe Roncalli de Paula Carneiro
felipecarneiro@unipac.br

O que vamos aprender nessa aula

- Importância de Download e Upload
- Segurança aplicado a Download e Upload
- Upload - Configurações
- Upload - Como evitar Injeção de arquivos
- Download - Implementação

Importância de Download e Upload

Troca de Dados com o Usuário:

- Download: Permite que os usuários obtenham informações, documentos ou mídia do servidor para seus dispositivos locais.
- Upload: Capacita os usuários a enviar dados, como documentos, imagens ou vídeos, para o servidor, contribuindo para a interatividade do sistema.

Importância de Download e Upload

Armazenamento e Compartilhamento de Informações:

- Download: Facilita o acesso a informações armazenadas no servidor, como relatórios, documentos e mídia.
- Upload: Possibilita que os usuários compartilhem informações e documentos, tornando-os acessíveis a outros usuários autorizados.

Importância de Download e Upload

Integração com Sistemas Externos:

- Download: Permite a integração de sistemas, como a obtenção de dados de APIs externas para enriquecer o conteúdo do sistema.
- Upload: Facilita a sincronização de dados gerados localmente com sistemas externos, contribuindo para uma experiência mais fluida.

Importância de Download e Upload

Gestão de Mídia e Conteúdo:

- Download: Fundamental para a exibição de imagens, vídeos, áudios e outros tipos de mídia no navegador do usuário.
- Upload: Possibilita que usuários contribuam com conteúdo multimídia, enriquecendo a experiência global do sistema.

Importância de Download e Upload

Backup e Recuperação de Dados:

- Download: Permite que usuários recuperem cópias de segurança de seus dados para evitar perdas.
- Upload: Facilita o processo de backup, permitindo que usuários enviem dados críticos para armazenamento seguro.

Importância de Download e Upload

Backup e Recuperação de Dados:

- Download: Permite que usuários recuperem cópias de segurança de seus dados para evitar perdas.
- Upload: Facilita o processo de backup, permitindo que usuários enviem dados críticos para armazenamento seguro.

Importância de Download e Upload

Funcionalidades de E-Commerce:

- Download: Essencial para o acesso a faturas, recibos e informações sobre produtos.
- Upload: Permite que usuários enviem documentos necessários, como comprovantes de pagamento ou formulários.

Importância de Download e Upload

Download e upload desempenham papéis cruciais na construção de sistemas web modernos, influenciando diretamente a interatividade, colaboração, comunicação e eficiência global do sistema. Entender e implementar essas funcionalidades de maneira segura e eficaz é essencial para o sucesso de muitas aplicações web.

Segurança aplicado a Download e Upload

Validação de Tipo de Arquivo:

- Desafio: Garantir que apenas tipos de arquivos permitidos sejam enviados ou baixados.
- Considerações de Segurança: Utilizar verificação de tipo MIME e extensão de arquivo. Evitar depender exclusivamente da extensão do arquivo fornecida pelo cliente.

Tamanho de Arquivo:

- Desafio: Gerenciar o tamanho dos arquivos para evitar ataques de negação de serviço.
- Considerações de Segurança: Limitar o tamanho máximo de upload. Realizar validação tanto no lado do cliente quanto no lado do servidor.

Segurança aplicado a Download e Upload

Prevenção de Injeção de Arquivos:

- Desafio: Evitar que usuários maliciosos enviem arquivos contendo código malicioso.
- Considerações de Segurança: Renomear os arquivos enviados, utilizar sanitização de nomes de arquivo, e evitar o uso de nomes dinâmicos fornecidos pelo usuário.

Proteção contra Ataques de Força Bruta:

- Desafio: Impedir ataques que visam adivinhar ou forçar senhas de acesso aos arquivos.
- Considerações de Segurança: Implementar mecanismos de controle de acesso, como autenticação e autorização adequadas. Monitorar e bloquear tentativas de acesso suspeitas.

Segurança aplicado a Download e Upload

Exposição de Dados Sensíveis:

- Desafio: Garantir que arquivos sensíveis não sejam acessíveis por usuários não autorizados.
- Considerações de Segurança: Implementar controle de acesso granular. Armazenar arquivos sensíveis fora do diretório web root. Validar a autenticação antes de conceder acesso.

Autenticação e Autorização Adequadas:

- Desafio: Garantir que apenas usuários autorizados possam realizar operações de upload e download.
- Considerações de Segurança: Implementar autenticação robusta. Atribuir permissões específicas para operações de upload e download. Monitorar atividades suspeitas.

Segurança aplicado a Download e Upload

A segurança em operações de upload e download de arquivos é crucial para proteger os sistemas contra uma variedade de ameaças. Considerações cuidadosas devem ser aplicadas em todas as fases, desde a validação dos dados até a implementação de controles de acesso e monitoramento proativo. Uma abordagem em camadas, combinando práticas de codificação seguras e tecnologias de segurança, é essencial para mitigar riscos e garantir a integridade e confidencialidade dos dados.

Upload - Configurações

A verificação das configurações do PHP para a manipulação de arquivos é uma etapa importante para garantir que o ambiente esteja adequadamente configurado e seguro.

Certifique-se de que as configurações são ajustadas conforme necessário no arquivo php.ini do seu servidor. Após fazer alterações a seguir, é recomendável reiniciar o servidor web para que as configurações tenham efeito.

Upload - Configurações

Configuração de Upload de Arquivos:

- file_uploads: Certifique-se de que esta configuração está habilitada (file_uploads = On). Isso permite o upload de arquivos via HTTP.
- upload_max_filesize: Define o tamanho máximo permitido para o upload de arquivos. Ajuste conforme necessário (upload_max_filesize = 20M para 20 megabytes, por exemplo).

Upload - Configurações

Configuração de Manipulação de Arquivos:

- post_max_size: Deve ser maior ou igual a upload_max_filesize para permitir uploads bem-sucedidos.
- max_execution_time: Define o tempo máximo (em segundos) que um script PHP pode ser executado. Ajuste conforme necessário para processar uploads mais longos.

Upload - Configurações

Configuração de Diretórios:

- upload_tmp_dir: Especifica o diretório temporário usado para armazenar arquivos durante o upload. Certifique-se de que o diretório tenha permissões adequadas.
- open_basedir: Se configurado, certifique-se de que os diretórios usados para upload e armazenamento estejam incluídos no open_basedir.

Upload - Configurações

Configurações de Segurança:

- allow_url_fopen: Deve ser desativado (allow_url_fopen = Off) para evitar possíveis ataques.
- allow_url_include: Também deve ser desativado (allow_url_include = Off) para evitar a inclusão remota de arquivos.

Upload - Exemplo

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Cadastro de Pessoa Física</title>
</head>
<body>

    <h2>Cadastro de Pessoa Física</h2>

    <form action="processa_cadastro.php" method="post" enctype="multipart/form-data">
        <label for="nome">Nome:</label>
        <input type="text" id="nome" name="nome" required><br>

        <label for="cpf">CPF:</label>
        <input type="text" id="cpf" name="cpf" required><br>

        <label for="data_nascimento">Data de Nascimento:</label>
        <input type="date" id="data_nascimento" name="data_nascimento" required><br>

        <label for="comprovante_endereco">Comprovante de Endereço:</label>
        <input type="file" id="comprovante_endereco" name="comprovante_endereco" accept=".pdf, .jpg, .jpeg, .png" required><br>

        <button type="submit">Cadastrar</button>
    </form>

</body>
</html>
```

Upload - Exemplo

- O atributo enctype="multipart/form-data" é necessário quando você está lidando com o envio de arquivos em formulários.
- O campo de CPF usa o atributo pattern para garantir que o formato seja respeitado.
- O campo de data de nascimento usa o elemento input com o tipo date.
- O campo de comprovante de endereço é um campo de arquivo (input type="file") e aceita apenas arquivos com extensões específicas (.pdf, .jpg, .jpeg, .png).

Upload - Exemplo

```php
<?php
// Verifica se o formulário foi enviado
if ($_SERVER["REQUEST_METHOD"] === "POST") {
    // Obtém os dados do formulário
    $nome = $_POST["nome"];
    $cpf = $_POST["cpf"];
    $data_nascimento = $_POST["data_nascimento"];
    // Upload do comprovante de endereço
    $upload_dir = "uploads/"; // Diretório para salvar os uploads
    $comprovante_endereco = $_FILES["comprovante_endereco"];
    // Verifica se o upload foi bem-sucedido
    if ($comprovante_endereco["error"] === UPLOAD_ERR_OK) {
        $comprovante_endereco_nome = $upload_dir . basename($comprovante_endereco["name"]);
        move_uploaded_file($comprovante_endereco["tmp_name"], $comprovante_endereco_nome);
    } else {
        // Trate erros de upload aqui
        echo "Erro no upload do comprovante de endereço.";
    }
    // Agora você pode realizar ações adicionais, como salvar os dados em um banco de dados
    // Exemplo: exibindo os dados para confirmação
    echo "<h2>Dados Recebidos:</h2>";
    echo "<p><strong>Nome:</strong> $nome</p>";
    echo "<p><strong>CPF:</strong> $cpf</p>";
    echo "<p><strong>Data de Nascimento:</strong> $data_nascimento</p>";
    echo "<p><strong>Comprovante de Endereço:</strong> $comprovante_endereco_nome</p>";
} else {
    // Se alguém acessar este script diretamente, redirecione para o formulário
    header("Location: formulario_cadastro.php");
    exit();
}
?>
```

Upload - Exemplo

- Este script verifica se o formulário foi enviado ($_SERVER["REQUEST_METHOD"] === "POST").
- Obtém os dados do formulário (nome, CPF, data de nascimento).
- Realiza o upload do comprovante de endereço para o diretório especificado (uploads/).
- Exibe os dados recebidos para confirmação. Neste ponto, você pode realizar ações adicionais, como salvar os dados em um banco de dados.
- Se alguém tentar acessar este script diretamente, é redirecionado de volta para o formulário de cadastro.

Upload

- Podemos adicionar verificações para deixar nosso código ainda mais seguro, como a verificação do tipo MIME e validação do tamanho de arquivo.

```php
// Verifica o tipo do arquivo (apenas PDF)
$allowed_types = ['application/pdf'];
$file_info = finfo_open(FILEINFO_MIME_TYPE);
$mime_type = finfo_file($file_info, $comprovante_endereco["tmp_name"]);
finfo_close($file_info);

if (!in_array($mime_type, $allowed_types)) {
    echo "Erro: O arquivo deve ser um PDF.";
    exit();
}
```

```php
// Verifica o tamanho do arquivo (menos de 30MB)
if ($comprovante_endereco["size"] > 30 * 1024 * 1024) {
    echo "Erro: O arquivo deve ter menos de 30MB.";
    exit();
}
```

Upload

- Podemos adicionar verificações para deixar nosso código ainda mais seguro, como a verificação do tipo MIME e validação do tamanho de arquivo.

```php
// Verifica o tipo do arquivo (apenas PDF)
$allowed_types = ['application/pdf'];
$file_info = finfo_open(FILEINFO_MIME_TYPE);
$mime_type = finfo_file($file_info, $comprovante_endereco["tmp_name"]);
finfo_close($file_info);

if (!in_array($mime_type, $allowed_types)) {
    echo "Erro: O arquivo deve ser um PDF.";
    exit();
}
```

```php
// Verifica o tamanho do arquivo (menos de 30MB)
if ($comprovante_endereco["size"] > 30 * 1024 * 1024) {
    echo "Erro: O arquivo deve ter menos de 30MB.";
    exit();
}
```

Upload - Como evitar Injeção de arquivos

A injeção de arquivos é uma ameaça potencialmente séria que pode ocorrer quando um aplicativo web aceita e processa arquivos enviados pelos usuários de maneira inadequada. Esses ataques podem levar à execução não autorizada de código, revelação de informações sensíveis ou danificação do sistema.

Upload - Como evitar Injeção de arquivos

Validação de Tipo de Arquivo:

- Ação: Verificar se o tipo MIME do arquivo corresponde ao tipo esperado.
- Implementação: Utilize bibliotecas ou funções específicas para verificar o tipo de arquivo, e não dependa apenas da extensão do arquivo fornecida pelo usuário.

Restrições de Tamanho:

- Ação: Limitar o tamanho dos arquivos aceitos.
- Implementação: Configure limites no servidor para o tamanho máximo de upload e processe apenas arquivos dentro desses limites.

Upload - Como evitar Injeção de arquivos

Renomeação de Arquivos:

- Ação: Renomear os arquivos enviados para evitar execução de código a partir do nome do arquivo.
- Implementação: Ao salvar o arquivo, renomeie-o para um nome seguro, evitando caracteres especiais e mantendo apenas a extensão do arquivo.

Isolamento de Arquivos:

- Ação: Armazenar os arquivos enviados fora do diretório web root.
- Implementação: Certifique-se de que os arquivos carregados não possam ser executados diretamente pelo servidor web, evitando a execução de scripts maliciosos.

Upload - Como evitar Injeção de arquivos

Autenticação e Autorização Adequadas:

- Ação: Garantir que apenas usuários autenticados e autorizados possam realizar operações de upload e download.
- Implementação: Implementar um sistema de controle de acesso robusto, com permissões específicas para operações relacionadas a arquivos.

Validação de Dados:

- Ação: Validar todos os dados fornecidos pelos usuários, incluindo os nomes dos arquivos.
- Implementação: Utilizar funções de validação para garantir que os nomes de arquivos não contenham caracteres ou sequências suspeitas.

Upload - Como evitar Injeção de arquivos

A implementação dessas medidas ajuda a reduzir significativamente o risco de ataques de injeção de arquivos, fortalecendo a segurança do sistema web. A abordagem mais eficaz é adotar uma mentalidade de segurança desde o início do desenvolvimento e manter práticas de segurança ao longo do ciclo de vida do aplicativo.

Download - Implementação

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Listagem de Arquivos</title>
</head>
<body>
    <h2>Listagem de Arquivos no Servidor</h2>
    <?php
    $diretorio = 'caminho/do/seu/diretorio'; // Substitua pelo caminho do seu diretório
    $arquivos = scandir($diretorio);

    // Remove os diretórios . e ..
    $arquivos = array_diff($arquivos, array('..', '.'));

    if (count($arquivos) > 0) {
        echo "<ul>";
        foreach ($arquivos as $arquivo) {
            echo "<li><a href=\"download.php?arquivo=" . urlencode($arquivo) . "\">" . $arquivo . "</a></li>";
        }
        echo "</ul>";
    } else {
        echo "<p>Nenhum arquivo encontrado.</p>";
    }
    ?>
</body>
</html>
```

Download - Implementação

- Utiliza a função scandir para obter a lista de arquivos no diretório.
- Remove os diretórios ., .. do array resultante.
- Gera links para download de cada arquivo, apontando para um arquivo chamado download.php que criaremos a seguir.

Download - Implementação

```php
<?php
if (isset($_GET['arquivo'])) {
    $arquivo = $_GET['arquivo'];
    $caminho = 'caminho/do/seu/diretorio/' . $arquivo; // Substitua pelo caminho do seu diretório

    if (file_exists($caminho)) {
        // Define os cabeçalhos para download
        header('Content-Description: File Transfer');
        header('Content-Type: application/octet-stream');
        header('Content-Disposition: attachment; filename="' . basename($caminho) . '"');
        header('Content-Transfer-Encoding: binary');
        header('Expires: 0');
        header('Cache-Control: must-revalidate');
        header('Pragma: public');
        header('Content-Length: ' . filesize($caminho));

        // Lê o arquivo e o envia para o cliente
        readfile($caminho);
        exit();
    } else {
        echo "Arquivo não encontrado.";
    }
} else {
    echo "Parâmetro 'arquivo' não fornecido.";
}
?>
```

Download - Implementação

- O arquivo download.php recebe o nome do arquivo através do parâmetro da URL ($_GET['arquivo']).
- Usa a função header para definir os cabeçalhos necessários para o download.
- Utiliza a função readfile para ler e enviar o conteúdo do arquivo para o cliente.

Dúvidas??