Guia de Vulnerabilidades na Aplicação ChitChat

1. Falta de Política de Senhas

Descrição:
Usuários podem definir senhas fracas como 1234 ou senha, comprometendo a segurança.

Exemplo:
Usuário define senha: 1234

Trecho vulnerável:

```
<?php
$senha = $_POST['senha'];
$query = "INSERT INTO usuarios (senha) VALUES ('$senha')";
mysqli_query($conexao, $query);
?>
```

Mitigação:
Defina regras de complexidade mínima (tamanho, caracteres especiais) e implemente políticas de expiração.

2. Tratamento de Erro Inapropriado

Descrição:
Mensagens de erro detalhadas revelam informações internas do sistema.

Exemplo:
```
Erro ao conectar: usuário root@localhost não encontrado
```

Trecho vulnerável:

```
<?php
$conexao = mysqli_connect("localhost", "root", "senha");
if (!$conexao) {
   die(mysqli_error($conexao));
}
```

```
?>
```

Mitigação:
Exiba mensagens genéricas para o usuário e registre os detalhes técnicos apenas em logs internos.

3. Falta de Proteção a Ataques de Força Bruta

Descrição:
Sem limite de tentativas, atacantes podem testar milhares de senhas.

Exemplo:
Login tenta 1000 senhas em 1 minuto.

Trecho vulnerável:

```
<?php
if ($_POST['user'] == 'admin' && $_POST['pass'] == '1234') {
   echo "Bem-vindo, admin!";
}
?>
```

Mitigação:
Implemente bloqueio temporário após várias tentativas falhas e utilize CAPTCHA.

4. Armazenamento de Senhas em Claro

Descrição:
Senhas salvas sem criptografia podem ser expostas facilmente.

Exemplo:
Tabela de usuários contém: `admin | 1234`

Trecho vulnerável:

```
<?php
$senha = $_POST['senha'];
$query = "INSERT INTO usuarios (senha) VALUES ('$senha')";
```

```
mysqli_query($conexao, $query);
?>
```

Mitigação:
Armazene senhas utilizando algoritmos de hash robustos, como bcrypt ou Argon2.

5. XSS (Cross-Site Scripting) Reflected

Descrição:
Scripts injetados retornam na resposta ao usuário.

Exemplo:
URL: http://site.com?q=<script>alert(1)</script>

Trecho vulnerável:

```
<?php
echo "Você pesquisou: " . $_GET['q'];
?>
```

Mitigação:
Sanitize e escape todas as entradas antes de exibi-las no HTML (ex.: `htmlspecialchars` no PHP).

6. SQL Injection

Descrição:
Entrada do usuário manipulada permite executar comandos SQL arbitrários.

Exemplo:
Login: `' OR '1'='1`

Trecho vulnerável:

```
<?php
$user = $_POST['user'];
$pass = $_POST['pass'];
$query = "SELECT * FROM usuarios WHERE user='$user' AND
```

```
pass='$pass'";
$result = mysqli_query($conexao, $query);
?>
```

Mitigação:
Use consultas preparadas (prepared statements) e nunca concatene entradas diretamente em SQL.

7. Unrestricted File Upload

Descrição:
Usuário faz upload de scripts maliciosos.

Exemplo:
Upload de `shell.php`

Trecho vulnerável:

```
<?php
move_uploaded_file($_FILES['arquivo']['tmp_name'], "uploads/" .
$_FILES['arquivo']['name']);
?>
```

Mitigação:
Restrinja os tipos de arquivo permitidos e armazene-os fora do diretório público do servidor.

8. File Inclusion (LFI/RFI)

Descrição:
Usuário pode incluir arquivos locais ou remotos.

Exemplo:
URL: `pagina.php?file=../../etc/passwd`

Trecho vulnerável:

```
<?php
include($_GET['file']);
?>
```

Mitigação:
Valide os arquivos incluídos com uma lista branca e desabilite `allow_url_include` no PHP.

9. Command Execution

Descrição:
Entrada do usuário é executada como comando no sistema.

Exemplo:
Input: `; rm -rf /`

Trecho vulnerável:

```
<?php
$arquivo = $_GET['arquivo'];
system("cat " . $arquivo);
?>
```

Mitigação:
Evite funções de execução de comandos (`system`, `exec`). Caso seja necessário, sanitize os parâmetros.

10. CSRF (Cross-Site Request Forgery)

Descrição:
Usuário é induzido a executar ações sem consentimento.

Exemplo:

```
<img src="http://banco.com/transferir?conta=9999&valor=1000">
```

Trecho vulnerável:

```
<img src="http://banco.com/transferir?conta=9999&valor=1000">
```

Mitigação:

Implemente tokens CSRF únicos por sessão e valide-os em cada requisição sensível.