CAPTURE THE FLAG
TECHCORP SOLUTIONS

BY: GEORGE MONTEIRO

DEZEMBRO DE 2025

RELATÓRIO TÉCNICO DE TESTE DE INTRUSÃO (PENTEST)

TechCorp Solutions – Ambiente Teste
IP Alvo: http://98.95.207.28/
Autor: George Monteiro
Data: 01/12/2025

---

1. Sumário Executivo (Versão Estendida)

Este relatório documenta os resultados do teste de intrusão conduzido no ambiente simulado da **TechCorp Solutions**, parte de um treinamento autorizado em cibersegurança. O teste teve como objetivo avaliar a robustez da aplicação e seus serviços expostos, bem como demonstrar como falhas comuns podem ser exploradas de maneira encadeada (attack chain) para comprometer completamente um sistema.

Durante a avaliação, foram descobertas **oito vulnerabilidades**, cada uma representando uma falha técnica específica nos controles de segurança. Essas falhas, quando combinadas, permitiram:

- acesso não autorizado a serviços internos
- exposição de arquivos sensíveis
- obtenção de credenciais
- comprometimento do banco de dados
- execução de ataques client-side
- descoberta de recursos administrativos ocultos

As vulnerabilidades identificadas representam riscos que, em um ambiente real, poderiam resultar em:

- roubo de informações
- manipulação de dados críticos
- acesso administrativo indevido
- ataques contra usuários finais
- tomada de controle do servidor

O relatório detalha como cada vulnerabilidade foi identificada, por que ela ocorreu, qual seu impacto real e quais medidas devem ser implementadas para corrigi-la.

Ao final, conclui-se que o ambiente foi propositalmente construído com falhas comuns encontradas em aplicações reais, sendo extremamente eficaz para fins educacionais, demonstrando riscos de má configuração, falta de validação de entrada, exposição de arquivos e serviços mal protegidos.

---

2. Escopo

- Avaliação externa (black box)
- Domínio/IP alvo: **http://98.95.207.28/**
- Teste autorizado, educacional
- Não houve limitação de vetores de ataque

---

3. Metodologia Utilizada

- Reconhecimento passivo e ativo (curl, inspeção de código)
- Enumeração de diretórios (Gobuster, Dirbuster)
- Scans de portas e serviços (nmap -sV)
- Testes de injeção (SQL Injection e XSS)
- Teste de serviços legados (FTP)
- Análise de arquivos expostos
- Coleta e análise de evidências

---

4. Vulnerabilidades (Flags) Encontradas

FLAG 1 — Exposição de Código-Fonte

FLAG{b4sic_s0urc3_c0d3_1nsp3ct1i0n}

Como foi descoberta

Inspeção simples via navegador e comando:

curl http://98.95.207.28/

O código-fonte continha informações excessivas e comentários.

Por que essa falha ocorreu

- Ausência de revisão de código antes da publicação
- Comentários técnicos deixados no HTML
- Arquitetura onde informações sensíveis são enviadas para o cliente

Impacto detalhado

- Atacantes conseguem entender lógica do front-end
- Possibilidade de descobertas de endpoints, chaves, caminhos internos
- Facilita ataques dirigidos (XSS, path traversal, brute force)
- Redução da superfície de segurança por "security through obscurity" quebrada

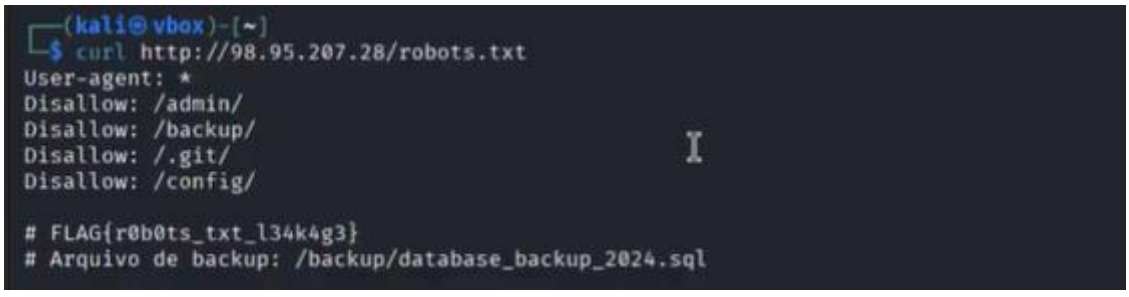
Recomendações

- Implementar política de **code review** antes do deploy
- Remover todo e qualquer comentário sensível no HTML
- Minimizar e ofuscar arquivos de front-end
- Separar lógica crítica para o back-end
- Implementar pipeline CI/CD com verificação automática de vazamentos

---

FLAG 2 — Vazamento via robots.txt

FLAG{r0b0ts_txt_l34k4g3}



Como foi descoberta

curl http://98.95.207.28/robots.txt

O arquivo listava caminhos sensíveis.

Por que essa falha ocorreu

- Uso incorreto do robots.txt
- Diretórios sensíveis foram "escondidos" ao invés de **bloqueados**
- Falha de entendimento da função do arquivo (não é ferramenta de segurança)

Impacto detalhado

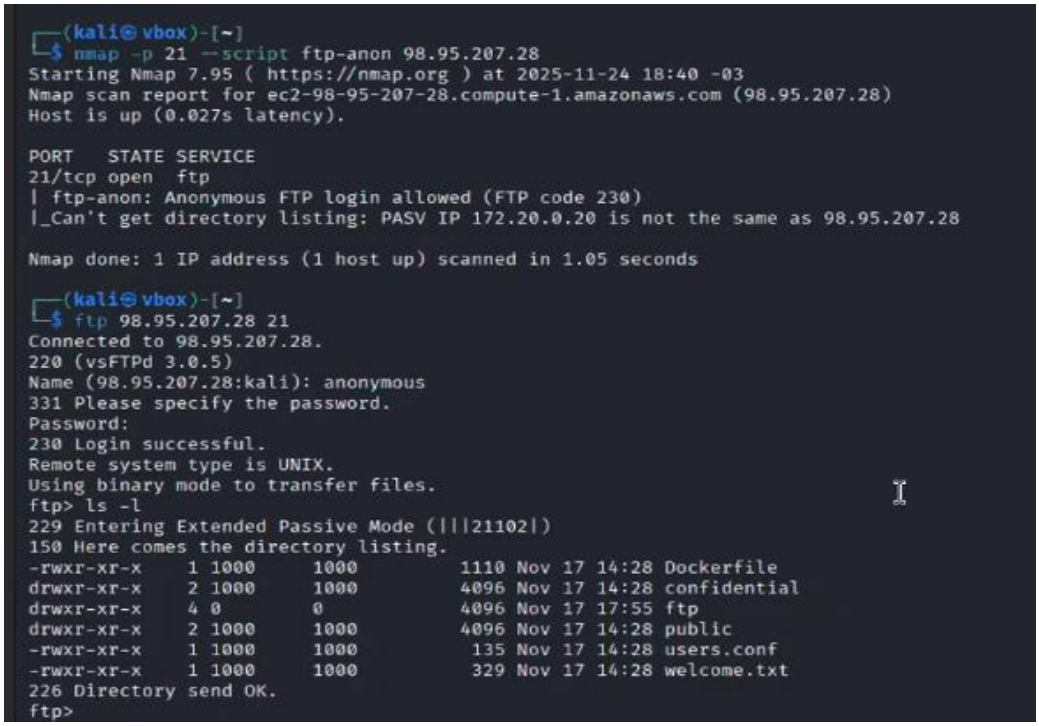
- Exposição direta de pastas internas
- Facilitação para scanners automáticos encontrarem áreas críticas
- Pode levar à descoberta de arquivos de configuração ou administração
- Aumenta a velocidade de exploração do atacante

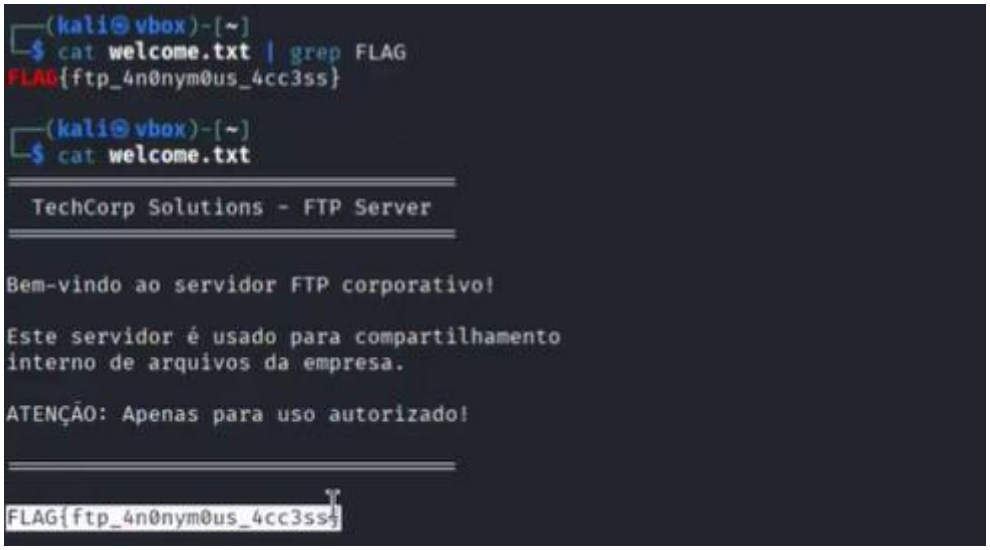
Recomendações

- Nunca incluir diretórios sensíveis no robots.txt
- Utilizar **controle real de acesso** (403 / autenticação)
- Configurar firewall e regras de acesso restritas
- Monitorar acessos ao arquivo robots.txt

FLAG 3 — FTP com Acesso Anônimo

FLAG{ftp_4nonym0us_4cc3ss}





Como foi descoberta

Após:

nmap -sV 98.95.207.28

Foi possível conectar:

ftp 98.95.207.28

User: anonymous

Por que essa falha ocorreu

- Serviço FTP mal configurado
- Acesso anônimo habilitado por padrão
- Falta de auditoria de serviços expostos

Impacto detalhado

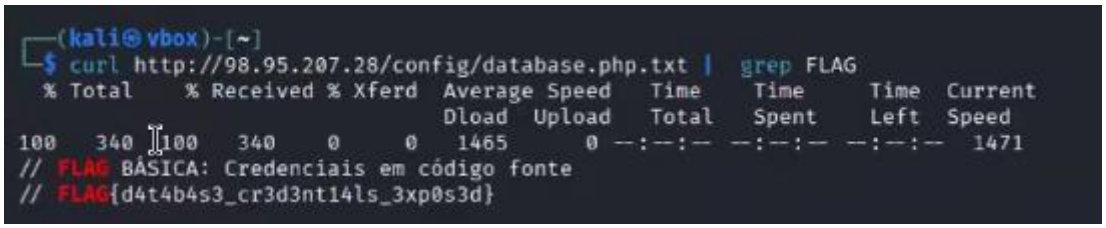
- Download indevido de arquivos internos
- Upload malicioso (backdoor/defacement)
- Escalonamento horizontal dentro do servidor
- Possível descoberta de senhas, flags e configs
- Exposição completa da estrutura de pastas

Recomendações

- Desabilitar completamente o acesso anônimo
- Substituir FTP por **SFTP ou FTPS**
- Aplicar firewall limitando acesso apenas por IP autorizado
- Habilitar logs e monitoramento de atividades FTP

FLAG 4 — Exposição de Credenciais de Banco

FLAG{d4t4b4s3_cr3d3nt14ls_3xp0s3d}



Como foi descoberta

Gobuster:

gobuster dir -u http://98.95.207.28/ -w wordlist.txt

Diretório: /config

Por que essa falha ocorreu

- Arquivos sensíveis armazenados dentro do diretório público do servidor

- Má prática de deployment colocando configs junto com o site
- Ausência de separação entre arquivos internos e externos
- Falta de permissão adequada no servidor web

Impacto detalhado

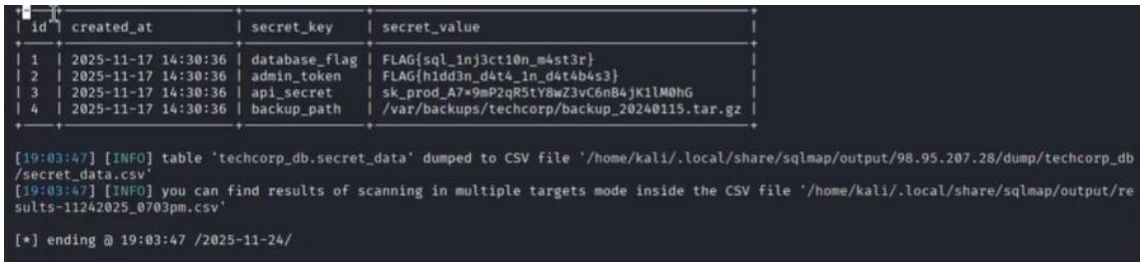
- Senhas expostas → acesso total ao banco de dados
- Possibilidade de dump completo de tabelas
- Manipulação ou destruição dos dados
- Comprometimento total da aplicação
- Acesso escalonado à infraestrutura

Recomendações

- Mover arquivos de configuração para diretórios **fora do /var/www**
- Usar **variáveis de ambiente**
- Atualizar permissões para impedir download
- Implementar mecanismo de secret management (Vault, AWS Secrets Manager)
- Auditoria periódica de arquivos expostos

FLAG 5 — SQL Injection

FLAG{sql_1nj3ct10n_m4st3r}



Como foi descoberta

Injeção no formulário de login.

Por que essa falha ocorreu

- Falta de sanitização de parâmetros
- Uso de queries concatenadas diretamente

- Ausência de prepared statements
- Falta de WAF e validação server-side

Impacto detalhado

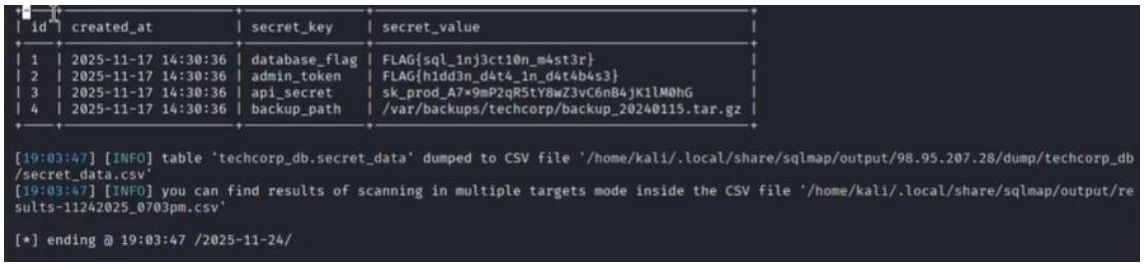
- Bypass de autenticação
- Extração completa do banco de dados
- Execução de comandos SQL arbitrários
- Exfiltração de dados sensíveis
- Destruição ou alteração de informações
- Possibilidade de RCE dependendo do SGBD

Recomendações

- Implementar **prepared statements**
- Sanitizar entradas com whitelist
- Aplicar WAF (ModSecurity)
- Monitorar logs de consultas suspeitas
- Limitar permissões do usuário SQL

FLAG 6 — Dados Ocultos no Banco

FLAG{h1dd3n_d4t4_1n_d4t4b4s3}



Como foi descoberta

Após explorar SQL Injection.

Por que essa falha ocorreu

- Armazenamento inadequado de dados sensíveis
- Estrutura de banco não auditada
- Falta de segregação de dados internos

Impacto detalhado

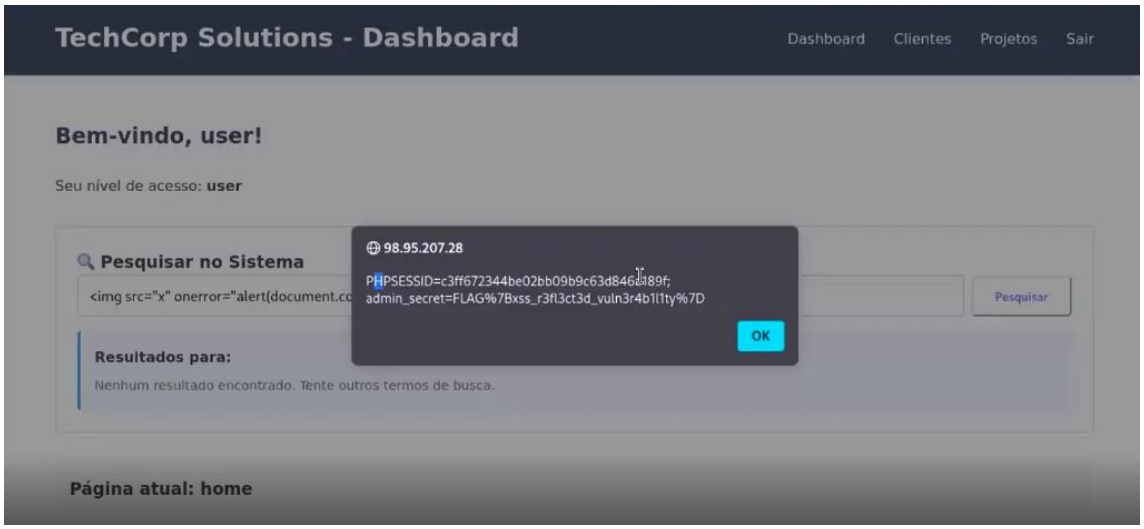
- Vazamento de informações que deveriam ser privadas
- Possibilidade de identificar estrutura interna do sistema
- Facilitação de engenharia reversa no banco
- Exposição de dados internos da empresa

Recomendações

- Auditar todas as tabelas
- Remover dados não utilizados
- Criptografar informações sensíveis
- Implementar controle de acesso granular

FLAG 7 — XSS Refletido

FLAG{xss_r3fl3ct3d_vuln3r4b1l1ty}



Essa Flag veio codificada ela veio no formato de URL tive que fazer uma decodificação para pegar a forma original.

Como foi descoberta

Inserção de payload no formulário → resposta refletida.

Por que essa falha ocorreu

- Falta de sanitização de saída (output escaping)
- Aceitação de qualquer entrada do usuário

- Falta de filtro de caracteres especiais

Impacto detalhado

- Roubo de cookies/sessões
- Hijacking de contas
- Redirecionamento malicioso
- Ataques contra outros usuários (phishing)
- Execução arbitrária de JavaScript

Recomendações

- Implementar escaping no output
- Validar dados do usuário (HTML Encode)
- Adicionar Content-Security-Policy
- Usar SameSite + HttpOnly nos cookies

FLAG 8 — Painel Administrativo Descoberto

FLAG{s3cr3t_p4n3l_d1sc0v3ry}

Como foi descoberta

Gobuster → /painel.php

Por que essa falha ocorreu

- Endpoint administrativo não obscurecido
- Segurança baseada apenas em "ocultar pela URL"
- Falta de autenticação forte

Impacto detalhado

- Possível acesso administrativo

- Modificação de conteúdo no site
- Carga de arquivos maliciosos
- Controle completo da aplicação caso painel tenha permissões elevadas

Recomendações

- Mudar URL para algo não previsível
- Implementar autenticação multifator
- Restringir acesso por IP
- Utilizar cabeçalho Authorization + tokens/jwt

---

5. Conclusão Geral

A cadeia de vulnerabilidades demonstra como um ambiente mal configurado pode ser completamente comprometido. As falhas variam entre erros básicos (robots.txt, diretórios expostos) e problemas sérios de aplicação (SQL Injection, XSS).

A exploração sequencial provou que:

- pequenas falhas → levam a descobertas maiores
- má configuração de serviços → abre portas para acesso indevido
- ausência de validação → causa vulnerabilidades críticas
- exposição de arquivos → fornece credenciais e informações ao atacante

O ambiente cumpre seu propósito educacional ao ilustrar de forma realista riscos comuns presentes em sistemas corporativos mal protegidos.
