PENTEST REPORT

Pentest Vantico App

Vantico
Attn. Kaique Bonato

São Paulo, August 10, 2025

Report Version: 1.0

CONFIDENTIAL

Table of Contents

1 Document Control

1.1 Team

Name	Contact	Role
João Zietolie Ciconet	me@joaocico.net	Pentester

2 Executive Summary

2.1 Overview

Este pentest teve como objetivo fornecer à equipe da VANTICO uma compreensão clara do estado atual de segurança do seu serviço, oferecendo recomendações para aprimorar sua resiliência contra possíveis ameaças cibernéticas. Nos testes foi possível obter cinco vulnerabilidades, sendo quatro de nível médio e uma de nível alto. Vale ressaltar que todas vulnerabilidades necessitam de uma conta válida na plataforma para que sejam exploradas e as vulnerabilidades de Cross-Site Scripting só puderam ser exploradas no ambiente de homologação, dada a presença de um Web Application Firewall (WAF) na aplicação de produção.

2.2 Identified Vulnerabilities

#	CVSS	Description	Page
H1	7.7	Fixação de Sessão entre Ambientes	8
M1	5.4	Reutilização de Identificador de Sessão	11
M2	4.7	Cross-site scripting armazenado no comentário de vulnerabilidade	14
M3	4.7	Cross-site scripting armazenado no nome da análise	17
M4	4.7	Cross-site scripting armazenado no nome da operação	21

Vulnerability Overview

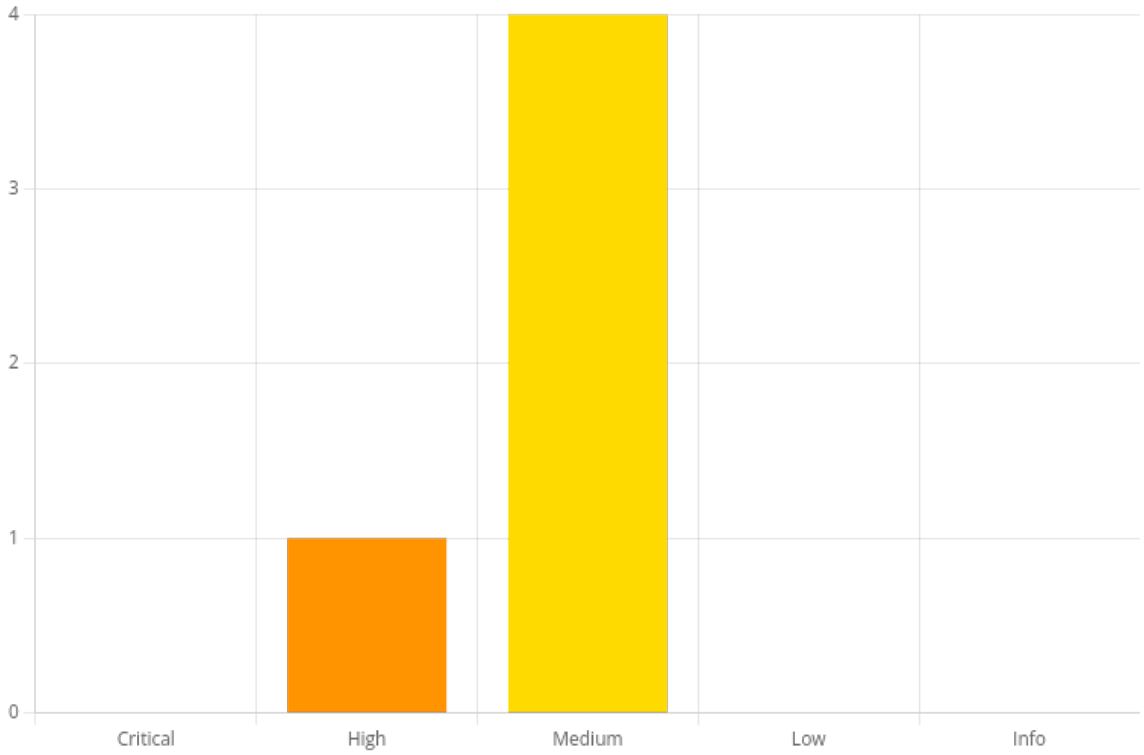

Figure 1 - Distribution of identified vulnerabilities

3 Methodology

O teste de intrusão foi conduzido seguindo a abordagem **Grey Box**, tendo acesso a contas de diversos níveis para a realização do teste.

Foram utilizadas como referência as diretrizes do **OWASP Web Security Testing Guide (WSTG)**, bem como boas práticas do **PTES (Penetration Testing Execution Standard)**, abrangendo as seguintes etapas:

1. **Coleta de Informações (Reconnaissance)**
    - Identificação de tecnologias e frameworks utilizados pela aplicação.
    - Análise de certificados digitais e metadados HTTP.
    - Enumeração de diretórios, endpoints e parâmetros acessíveis.
2. **Mapeamento da Aplicação**
    - Análise das funcionalidades acessíveis via interface web.
    - Identificação de pontos de entrada (forms, parâmetros GET/POST, cookies, headers).
    - Mapeamento de fluxos de autenticação e autorização.
3. **Análise de Vulnerabilidades**
    - Testes baseados no **OWASP Top 10 (2021)**, incluindo:
        - Injeções (SQLi, Command Injection, SSI).
        - Quebra de autenticação e gerenciamento de sessões.
        - Cross-Site Scripting (XSS).
        - Cross-Site Request Forgery (CSRF).
        - Exposição de dados sensíveis.
        - Server-Side Request Forgery (SSRF).
        - File Inclusion (LFI/RFI).
        - Path Traversal.
    - Testes adicionais de configuração e segurança de transporte (TLS, headers HTTP).
4. **Exploração Controlada**
    - Validação prática das vulnerabilidades identificadas.
    - Coleta de evidências (capturas de tela, requisições e respostas HTTP).
    - Testes limitados para evitar impactos no ambiente de produção.
5. **Pós-Exploração**
    - Análise de possíveis impactos.
    - Verificação de persistência e escalonamento de privilégios.
6. **Relatório e Recomendação**
    - Registro detalhado das vulnerabilidades encontradas, evidências, classificação de risco (CVSS 3.1) e recomendações de mitigação.
    - Entrega de relatório técnico e sumário executivo.

3.1 Objective

O objetivo do teste é buscar por falhas de segurança, especialmente as que possam vir a causar algum dano crítico para a organização e seus clientes.

3.2 Scope

System	Description
ptaas.vantico.com.br	HLG
app.vantico.com.br	PRD

3.3 User Accounts and Permissions

Provided Users

- gestao@vantico.com.br
- henrique.melo@vantico.com.brgestao@partner.com
- gestao@partner.com

4 Findings

H1: Fixação de Sessão entre Ambientes	
Score	7.7 (High)
Vector string	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:N/A:N
Target	• https://ptaas.vantico.com.br • https://app.vantico.com.br
References	https://owasp.org/www-community/attacks/Session_fixation

Overview

Foi identificada uma vulnerabilidade de Fixação de Sessão que permite que um identificador de sessão (PHPSESSID) gerado no ambiente de Homologação (ptaas) seja utilizado para obter acesso autenticado ao ambiente de Produção. Esta falha de isolamento permite que um usuário, incluindo um com privilégios elevados em um ambiente de menor segurança, acesse e manipule dados de produção, contornando os controles de autenticação específicos do ambiente produtivo.

Details

A aplicação permite que um cookie de sessão (PHPSESSID) criado e validado no ambiente de homologação seja aceito no ambiente de produção.

Causa Raiz Provável: A causa mais provável é o compartilhamento do mesmo repositório de armazenamento de sessões (ex: um diretório no servidor, Redis ou Memcached) entre os servidores de produção e homologação. Ao receber um PHPSESSID, o ambiente de produção localiza o arquivo de sessão correspondente no repositório compartilhado e, como os dados da sessão (email, permission, etc.) são estruturalmente idênticos, considera a sessão válida, concedendo acesso indevido.

Passos para Reprodução:

1. Autenticar com um usuário qualquer (ex: gestao@vantico.com.br) no ambiente de Homologação (https://ptaas.dominio.com).
2. Utilizando as ferramentas de desenvolvedor do navegador, copiar o valor do cookie PHPSESSID gerado para a sessão de homologação.
3. Abrir o ambiente de Produção (https://app.dominio.com) em uma aba anônima ou com os cookies limpos.
4. Injetar o cookie PHPSESSID copiado no passo 2 nas ferramentas de desenvolvedor para o domínio de produção.
5. Atualizar a página do ambiente de Produção.

Resultado Observado: O usuário é autenticado no ambiente de Produção com a identidade e as permissões da sessão originada em Homologação, obtendo acesso direto aos dados e funcionalidades de produção.

Estando com os dois ambientes autenticados com credenciais válidas, copiamos o cookie de sessão do usuário do ambiente HLG e alteramos diretamente no navegador do usuário conectado em PRD, após atualizar a página pode-se observar vários dados difusos, dentre eles a lista de clientes da VANTICO e vulnerabilidades reportadas em outros escopos:

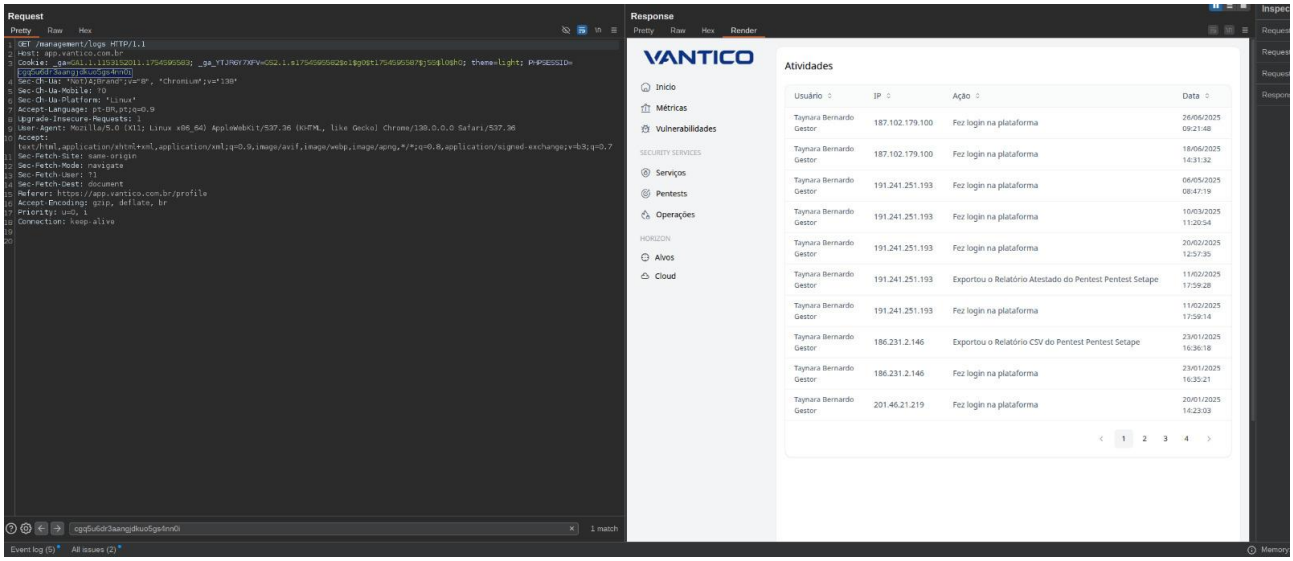

Figura 1 - Dados de logs obtidos

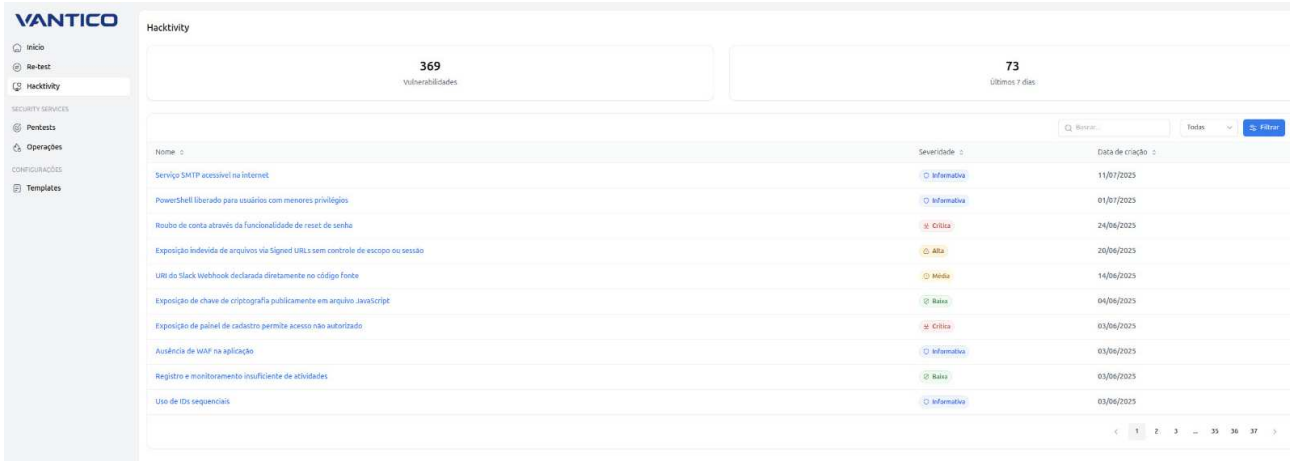

Figura 2 - Lista de vulnerabilidades que podem ser visualizadas

Esta vulnerabilidade traz os seguintes impactos:

1. Comprometimento de Dados de Produção: Um invasor com acesso a uma conta em homologação (que geralmente possui controles de segurança mais frouxos) pode acessar, modificar ou excluir dados sensíveis de clientes no ambiente de produção.
2. Escalação de Privilégios Horizontal e Vertical: Um usuário de baixo privilégio em homologação pode se passar por um usuário de alto privilégio em produção se os e-mails coincidirem, ou simplesmente acessar dados de produção que não deveria.

3. Quebra de Conformidade e Confiança: A falha em isolar os ambientes viola princípios básicos de segurança da informação e pode levar a não conformidade com regulações como a LGPD, além de erodir a confiança dos clientes.
4. Risco de Sequestro de Sessão: A reutilização do ID de sessão aumenta a janela de oportunidade para ataques de sequestro de sessão (session hijacking), especialmente em ambientes com múltiplos usuários ou redes Wi-Fi públicas.

Recommendation

Correção Imediata:

- Isolar os Repositórios de Sessão: A medida mais importante é garantir que cada ambiente (Produção, Homologação, etc.) utilize um session.save_path completamente isolado e exclusivo. Isso deve ser configurado no php.ini ou na configuração do pool do PHP-FPM de cada servidor.
  - Exemplo (Produção): session.save_path = "/var/lib/php/sessions/prod"
  - Exemplo (Homologação): session.save_path = "/var/lib/php/sessions/homolog"
- Utilizar Nomes de Cookie de Sessão Distintos: Para evitar que o navegador envie o cookie de um ambiente para o outro, configure nomes diferentes para a sessão em cada ambiente. Código (no início do script): session_name('PRODSESSID'); para produção e session_name('HOMOLOGSESSID'); para homologação.

Correção Adicional (Boas Práticas):

- Regenerar o ID da Sessão no Login e Logout: Para corrigir a reutilização do ID, sempre regenere o identificador de sessão após qualquer alteração no estado de autenticação.
  - No Login (após validar as credenciais):

```
session_regenerate_id(true); // Invalida o ID antigo e gera um novo
$_SESSION['login'] = ...;
// ... resto do código de login
```

No Logout:

```
$_SESSION = array(); // Limpa os dados da sessão
if (ini_get("session.use_cookies")) {
    $params = session_get_cookie_params();
    setcookie(session_name(), '', time() - 42000,
        $params["path"], $params["domain"],
        $params["secure"], $params["httponly"]
     ); // Limpa o cookie do navegador
}
session_destroy(); // Destrói a sessão no servidor
```

M1: Reutilização de Identificador de Sessão	
Score	5.4 (Medium)
Vector string	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:N
Target	• https://ptaas.vantico.com.br/ • https://app.vantico.com.br/
References	https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/06-Session_Management_Testing/README

Overview

Foi observado um comportamento de Reutilização de Identificador de Sessão, onde, após o logout de um usuário X e o login subsequente de um usuário Y, o mesmo identificador de sessão é atribuído ao usuário Y. Isso indica que a sessão não é invalidada e regenerada corretamente, criando um risco de sequestro de sessão em cenários específicos.

Details

Foi observado que a aplicação não invalida o identificador de sessão (PHPSESSID) após o logout. Quando um novo usuário faz login no mesmo navegador, a aplicação reutiliza o PHPSESSID existente e apenas atualiza os dados da sessão no servidor.

Causa Raiz Provável: O código da aplicação não invoca a função session_regenerate_id(true) após uma mudança de estado de autenticação (login e logout). A função session_destroy() pode estar sendo chamada, o que apaga os dados, mas não necessariamente remove o cookie do navegador ou garante a geração de um novo ID. Passos para Reprodução:

1. Autenticar na aplicação com o Usuário X. Anotar o valor do PHPSESSID.
2. Realizar o logout do Usuário X.
3. Na mesma sessão do navegador, autenticar com o Usuário Y.
4. Verificar o valor do PHPSESSID.

Resultado Observado: O PHPSESSID do Usuário Y é o mesmo que foi utilizado pelo Usuário X.

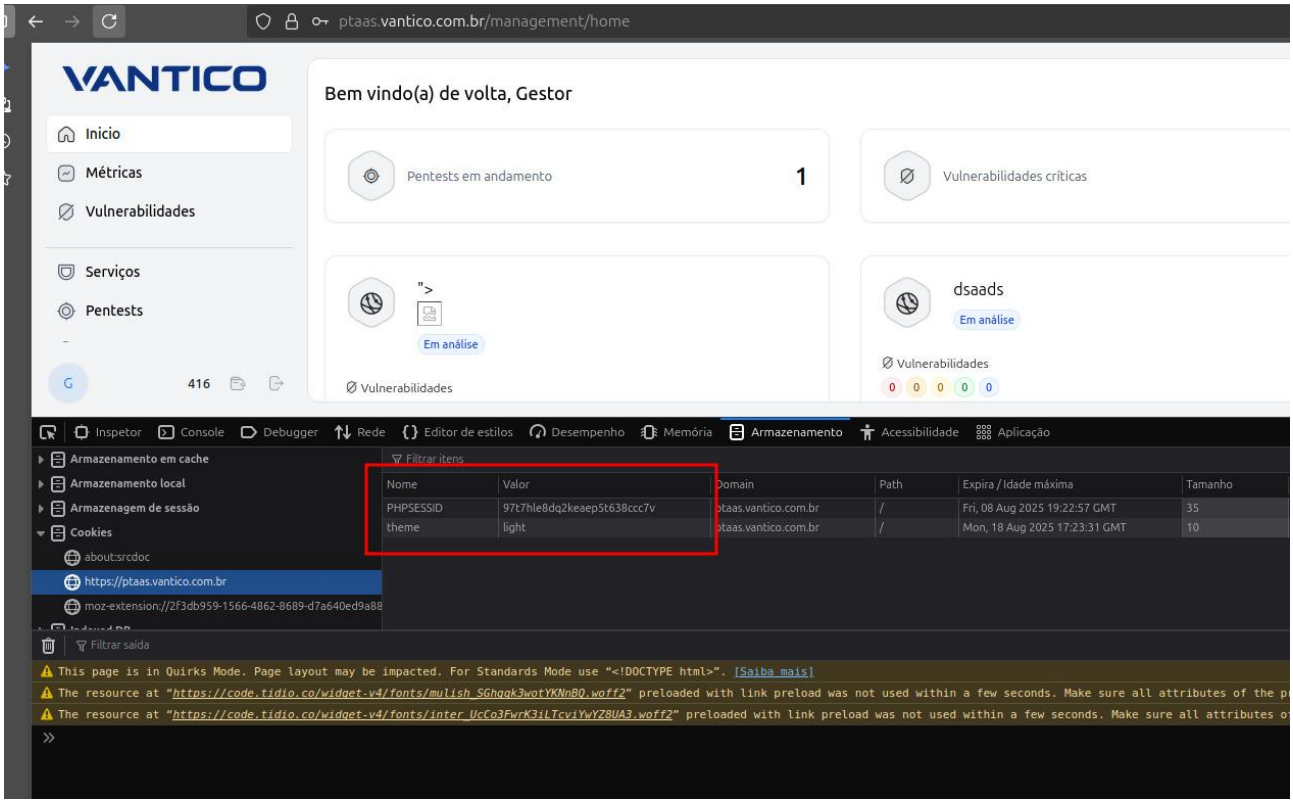

Figura 1 - Autenticado como Gestor

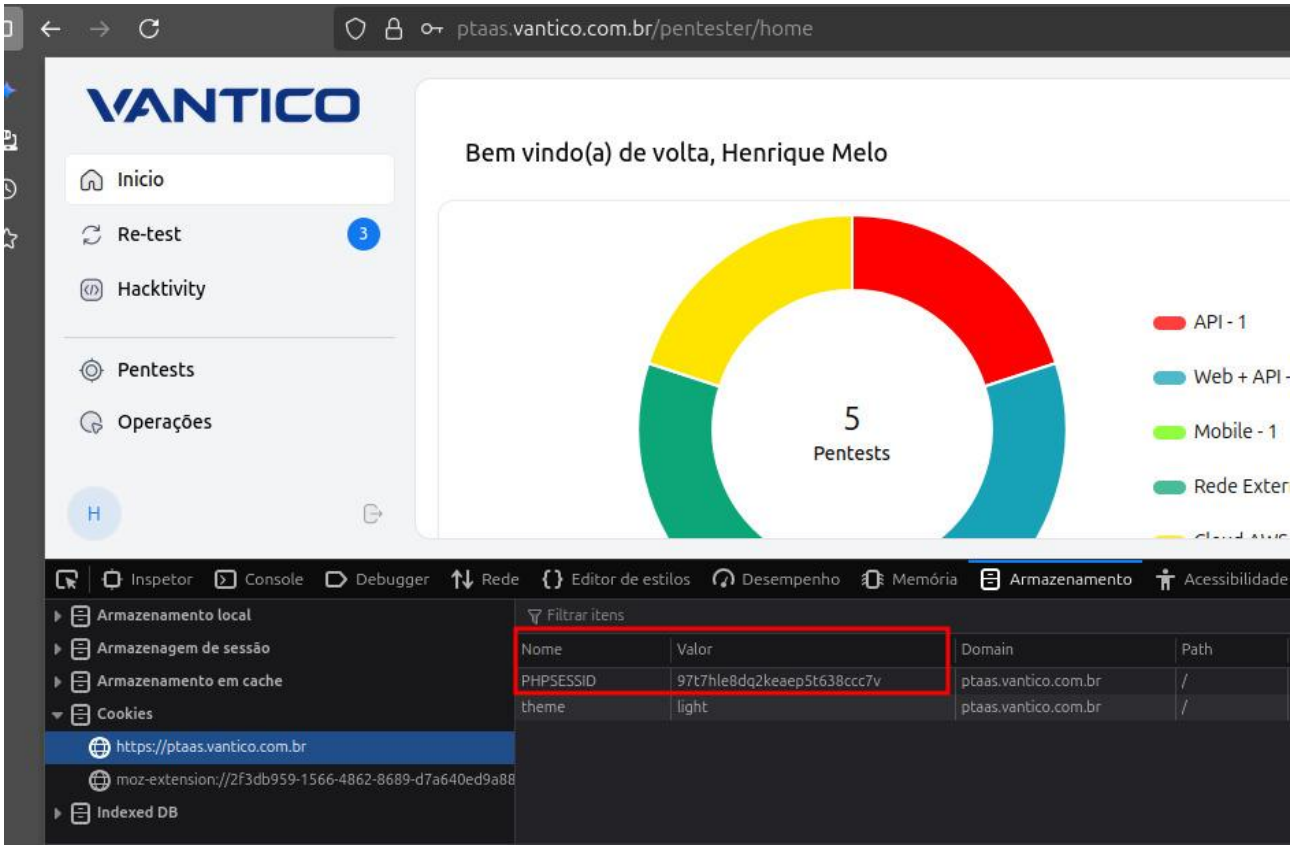

Figura 2 - Autenticado como Analista

Recommendation

Correção Imediata:

- Isolar os Repositórios de Sessão: A medida mais importante é garantir que cada ambiente (Produção, Homologação, etc.) utilize um session.save_path completamente isolado e exclusivo. Isso deve ser configurado no php.ini ou na configuração do pool do PHP-FPM de cada servidor.
  - Exemplo (Produção): session.save_path = "/var/lib/php/sessions/prod"
  - Exemplo (Homologação): session.save_path = "/var/lib/php/sessions/homolog"
- Utilizar Nomes de Cookie de Sessão Distintos: Para evitar que o navegador envie o cookie de um ambiente para o outro, configure nomes diferentes para a sessão em cada ambiente. Código (no início do script): session_name('PRODSESSID'); para produção e session_name('HOMOLOGSESSID'); para homologação.

Correção Adicional (Boas Práticas):

- Regenerar o ID da Sessão no Login e Logout: Para corrigir a reutilização do ID, sempre regenere o identificador de sessão após qualquer alteração no estado de autenticação.
  - No Login (após validar as credenciais):

```
session_regenerate_id(true); // Invalida o ID antigo e gera um novo
$_SESSION['login'] = ...;
// ... resto do código de login
```

No Logout:

```
$_SESSION = array(); // Limpa os dados da sessão
if (ini_get("session.use_cookies")) {
    $params = session_get_cookie_params();
    setcookie(session_name(), '', time() - 42000,
        $params["path"], $params["domain"],
        $params["secure"], $params["httponly"]
     ); // Limpa o cookie do navegador
}
session_destroy(); // Destrói a sessão no servidor
```

M2: Cross-site scripting armazenado no comentário de vulnerabilidade	
Score	4.7 (Medium)
Vector string	CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:U/C:L/I:L/A:L
Target	https://ptaas.vantico.com.br/
References	https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Overview

A vulnerabilidade Stored Cross-Site Scripting (XSS Armazenado) ocorre quando uma aplicação armazena permanentemente entradas maliciosas fornecidas por um usuário sem realizar a adequada sanitização, permitindo que códigos JavaScript maliciosos sejam executados sempre que a página for carregada. Ao contrário do XSS Refletido, que exige interação específica do usuário em uma única requisição, o XSS Armazenado afeta diversos usuários simultaneamente, uma vez que o conteúdo malicioso permanece armazenado e é executado sempre que alguém visualiza o dado comprometido. Tal vulnerabilidade pode ser explorada para o roubo de sessões, redirecionamento a páginas maliciosas, captura de teclas digitadas (keylogging) e outras ações que comprometem a segurança tanto dos usuários quanto da aplicação.

Details

Durante os testes de validação de dados na aplicação ptaas.vantico.com.br foi identificada a falta de validação de input do usuário no campo de comentário ao realizar a alteração do status de uma vulnerabilidade. A vulnerabilidade só foi possível de ser explorada pelo usuário Gestor, em que ao alterar o status de uma vulnerabilidade é possível adicionar scripts que serão posteriormente armazenados e renderizados pela aplicação.

A imagem abaixo demonstra o script sendo executado no campo de comentário:

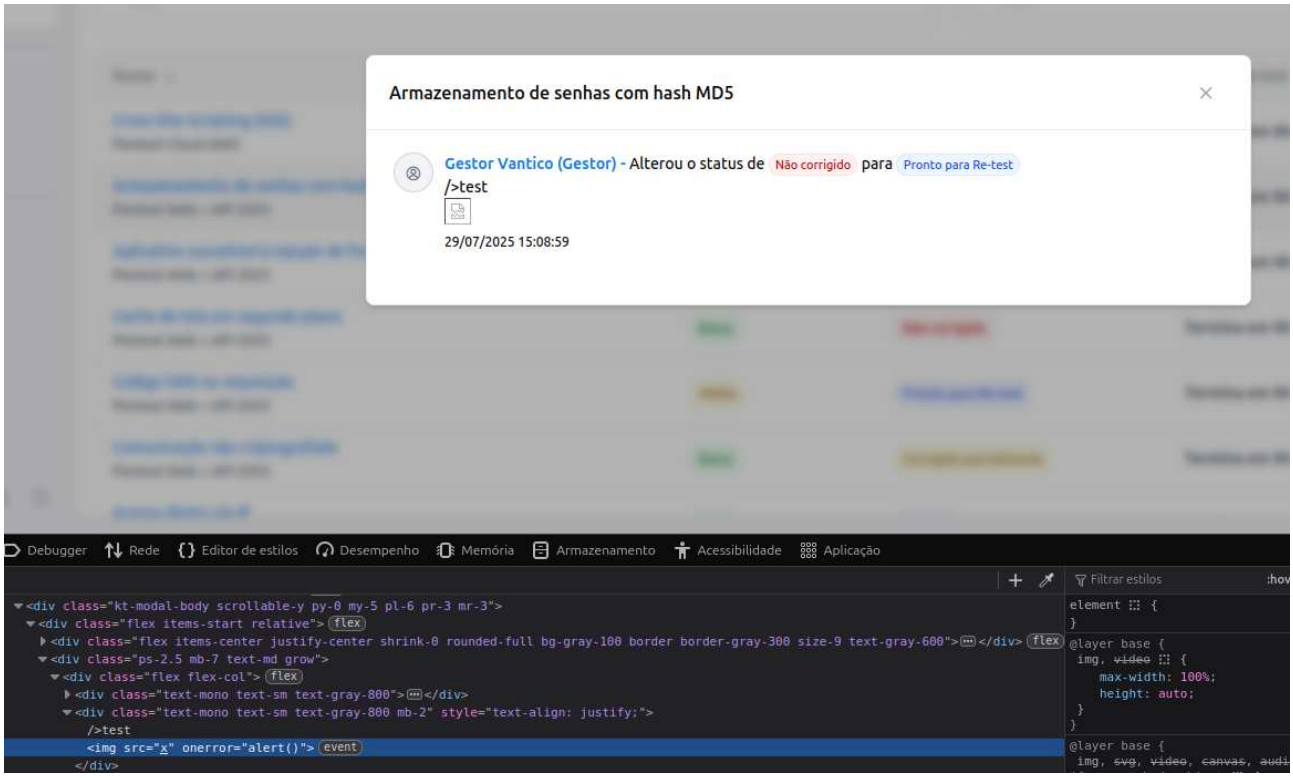

Figura 1 - Código sendo renderizado

O script também é executado em outras partes da aplicação em que é incluído, como na lista de apontamentos e no próprio apontamento.

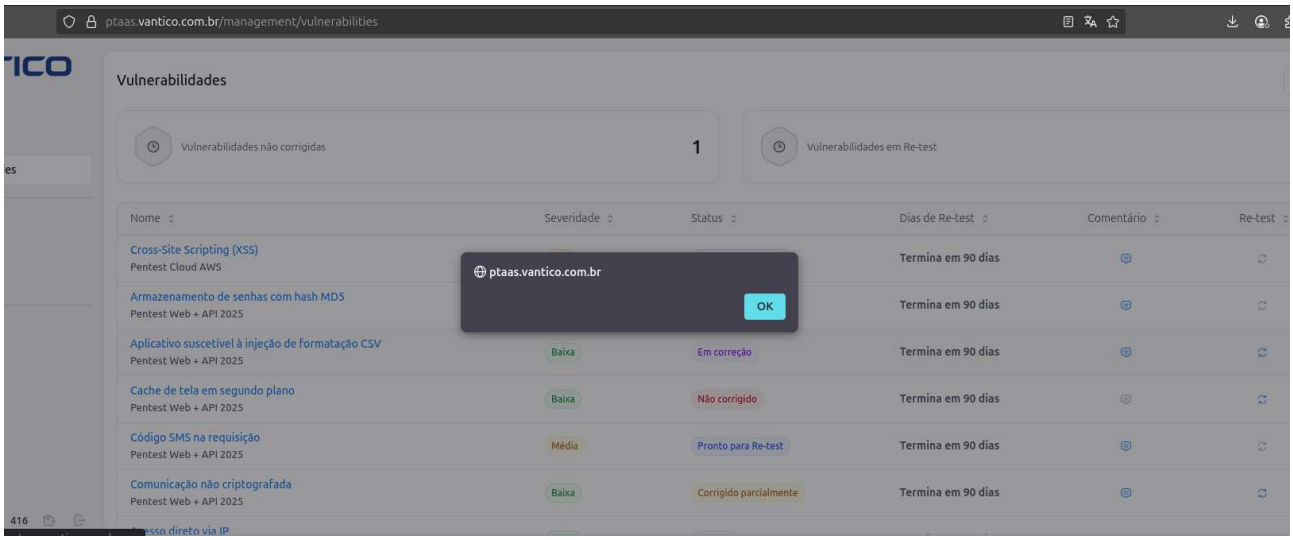

Figura 2 - Código executando na lista de apontamentos

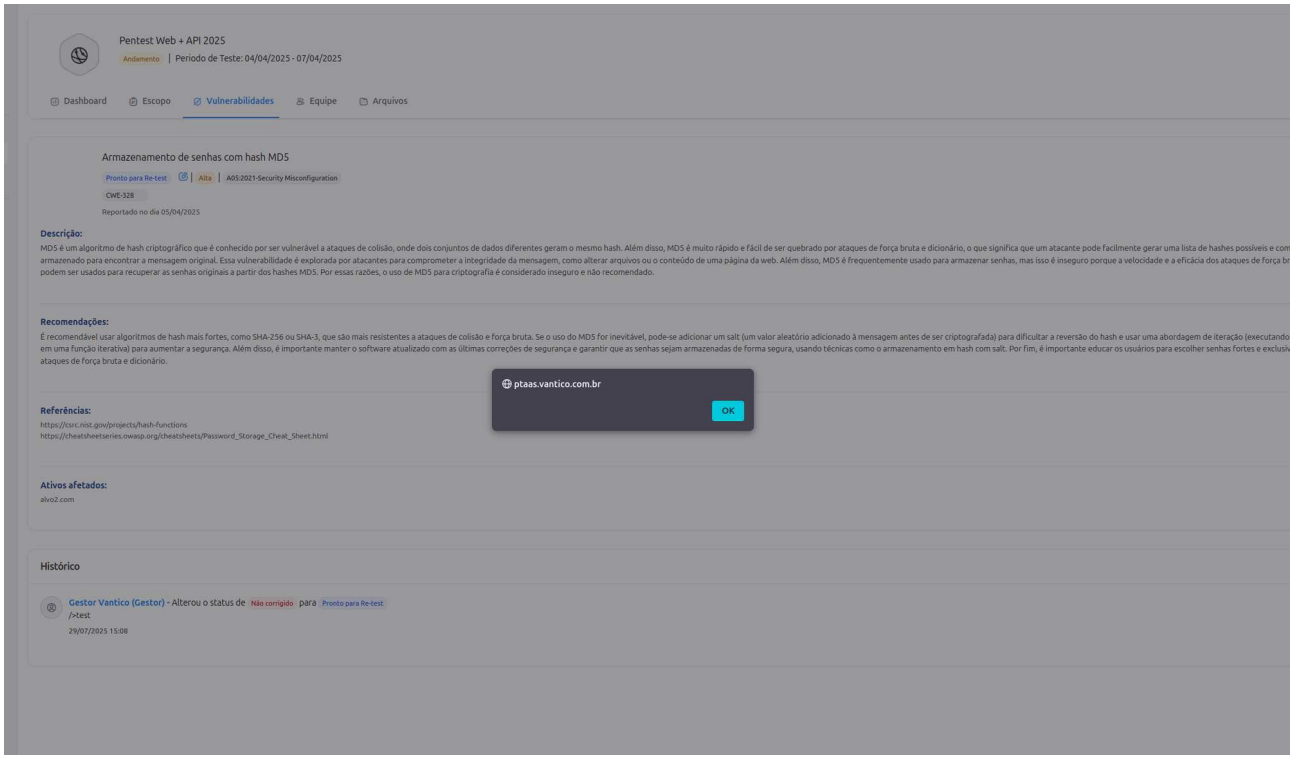

Figura 3 - Código executando dentro do próprio apontamento

Esta vulnerabilidade pode trazer impactos significativos, como por exemplo:

- **Comprometimento da Confiança do Usuário:** Usuários podem ser redirecionados para sites maliciosos ou ter suas contas comprometidas.
- **Violação de Conformidade:** Regulamentações como LGPD e GDPR podem ser infringidas se informações pessoais forem comprometidas.
- **Execução de Código Arbitrário no Navegador:** Um atacante pode executar comandos JavaScript maliciosos no contexto da vítima.
- **Roubo de Sessões de Usuários:** Cookies de autenticação podem ser extraídos e reutilizados por um invasor.
- **Modificação de Conteúdo da Página:** O atacante pode alterar a interface da aplicação para induzir usuários a interagirem com elementos falsificados.
- **Exfiltração de Dados Sensíveis:** Informações pessoais e credenciais podem ser capturadas por meio de scripts maliciosos.

Recommendation

- **Sanitização e Validação de Entradas:** Implemente uma rigorosa validação e sanitização de todas as entradas de dados, especialmente aquelas provenientes dos usuários. Garanta que apenas caracteres permitidos sejam aceitos e remova ou neutralize qualquer código malicioso.
- **Utilização de Content Security Policy (CSP):** Implemente uma política de Content Security Policy (CSP) eficaz para restringir o carregamento de recursos e scripts externos. Isso pode ajudar a prevenir a execução de scripts não autorizados.
- **Escape de Dados:** Ao exibir dados no lado do cliente, utilize técnicas de escape apropriadas para garantir que os dados sejam interpretados apenas como dados, não como código executável. Isso inclui o uso de funções de escape em frameworks e bibliotecas.

M3: Cross-site scripting armazenado no nome da análise	
Score	4.7 (Medium)
Vector string	CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:U/C:L/I:L/A:L
Target	https://ptaas.vantico.com.br/
References	https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Overview

A vulnerabilidade Stored Cross-Site Scripting (XSS Armazenado) ocorre quando uma aplicação armazena permanentemente entradas maliciosas fornecidas por um usuário sem realizar a adequada sanitização, permitindo que códigos JavaScript maliciosos sejam executados sempre que a página for carregada. Ao contrário do XSS Refletido, que exige interação específica do usuário em uma única requisição, o XSS Armazenado afeta diversos usuários simultaneamente, uma vez que o conteúdo malicioso permanece armazenado e é executado sempre que alguém visualiza o dado comprometido. Tal vulnerabilidade pode ser explorada para o roubo de sessões, redirecionamento a páginas maliciosas, captura de teclas digitadas (keylogging) e outras ações que comprometem a segurança tanto dos usuários quanto da aplicação.

Details

Durante os testes de validação de dados na aplicação "ptaas.vantico.com.br" foi identificada a falta de validação de input do usuário no campo de "nome" de análises. A vulnerabilidade só foi possível de ser explorada pelo usuário Gestor, pois requer permissão de alterar ou criar novos projetos.

A imagem abaixo demonstra o script sendo executado no campo de nome:

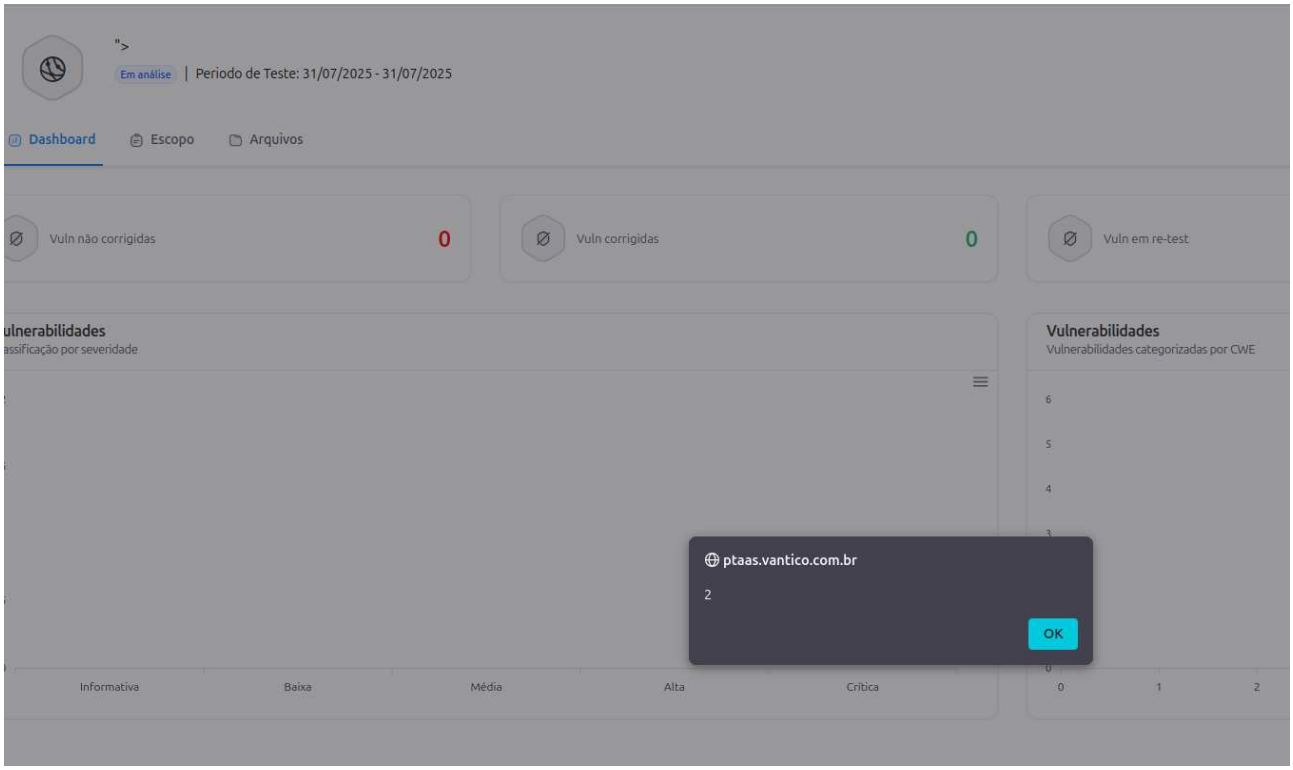

Figura 1 - Código sendo renderizado

O script também é executado em outras partes da aplicação em que é incluído, como na lista de apontamentos e no menu inicial.

Figura 2 - Código executando na lista de apontamentos

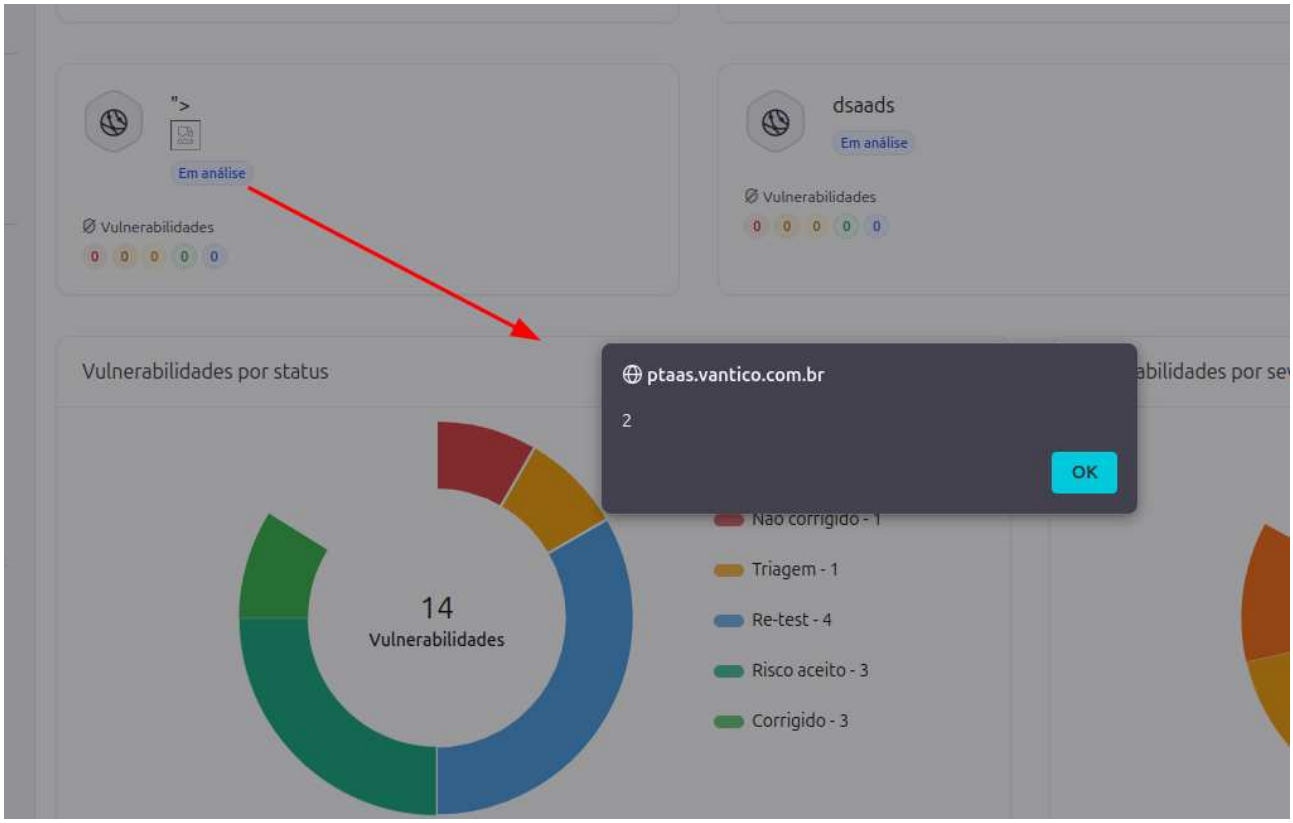

Figura 3 - Código executando no menu incial

Esta vulnerabilidade pode trazer impactos significativos, como por exemplo:

- **Comprometimento da Confiança do Usuário:** Usuários podem ser redirecionados para sites maliciosos ou ter suas contas comprometidas.
- **Violação de Conformidade:** Regulamentações como LGPD e GDPR podem ser infringidas se informações pessoais forem comprometidas.
- **Execução de Código Arbitrário no Navegador:** Um atacante pode executar comandos JavaScript maliciosos no contexto da vítima.
- **Roubo de Sessões de Usuários:** Cookies de autenticação podem ser extraídos e reutilizados por um invasor.
- **Modificação de Conteúdo da Página:** O atacante pode alterar a interface da aplicação para induzir usuários a interagirem com elementos falsificados.
- **Exfiltração de Dados Sensíveis:** Informações pessoais e credenciais podem ser capturadas por meio de scripts maliciosos.

Recommendation

- **Sanitização e Validação de Entradas:** Implemente uma rigorosa validação e sanitização de todas as entradas de dados, especialmente aquelas provenientes dos usuários. Garanta que apenas caracteres permitidos sejam aceitos e remova ou neutralize qualquer código malicioso.
- **Utilização de Content Security Policy (CSP):** Implemente uma política de Content Security Policy (CSP) eficaz para restringir o carregamento de recursos e scripts externos. Isso pode ajudar a prevenir a execução de scripts não autorizados.

- **Escape de Dados:** Ao exibir dados no lado do cliente, utilize técnicas de escape apropriadas para garantir que os dados sejam interpretados apenas como dados, não como código executável. Isso inclui o uso de funções de escape em frameworks e bibliotecas.

M4: Cross-site scripting armazenado no nome da operação	
Score	4.7 (Medium)
Vector string	CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:U/C:L/I:L/A:L
Target	https://ptaas.vantico.com.br/
References	https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Overview

A vulnerabilidade Stored Cross-Site Scripting (XSS Armazenado) ocorre quando uma aplicação armazena permanentemente entradas maliciosas fornecidas por um usuário sem realizar a adequada sanitização, permitindo que códigos JavaScript maliciosos sejam executados sempre que a página for carregada. Ao contrário do XSS Refletido, que exige interação específica do usuário em uma única requisição, o XSS Armazenado afeta diversos usuários simultaneamente, uma vez que o conteúdo malicioso permanece armazenado e é executado sempre que alguém visualiza o dado comprometido. Tal vulnerabilidade pode ser explorada para o roubo de sessões, redirecionamento a páginas maliciosas, captura de teclas digitadas (keylogging) e outras ações que comprometem a segurança tanto dos usuários quanto da aplicação.

Details

Durante os testes de validação de dados na aplicação "ptaas.vantico.com.br" foi identificada a falta de validação de input do usuário no campo de "nome" de operações. A vulnerabilidade só foi possível de ser explorada pelo usuário Gestor, pois requer permissão de alterar ou criar novoas operações.

A imagem abaixo demonstra o script sendo executado no campo de nome:

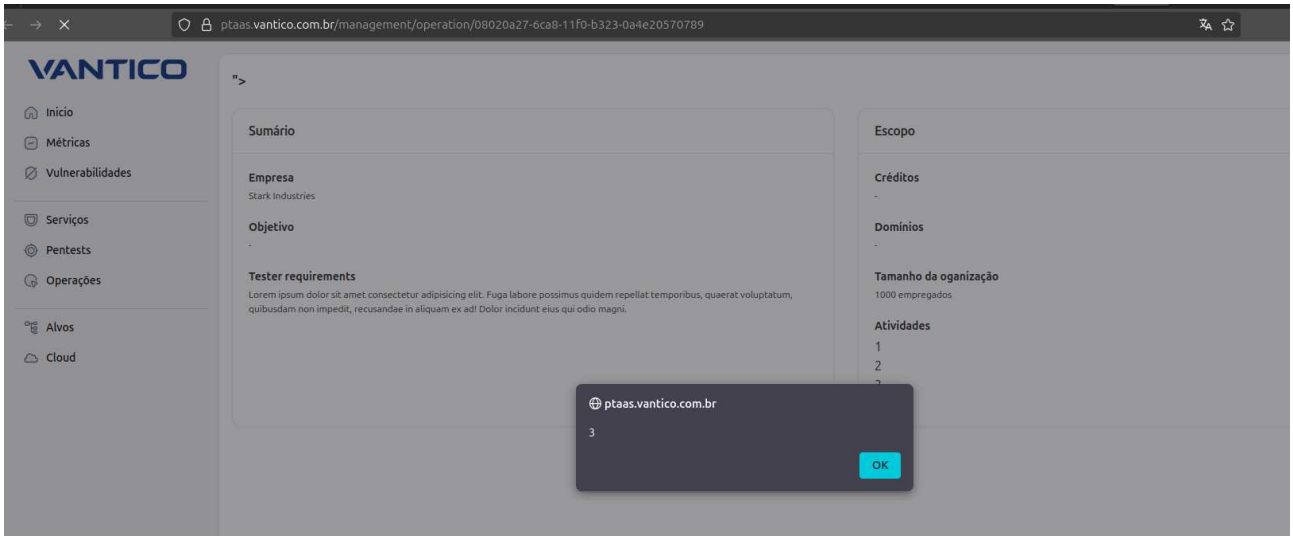

Figura 1 - Código sendo renderizado

O script também é executado em outras partes da aplicação em que é incluído, como na lista de operações:

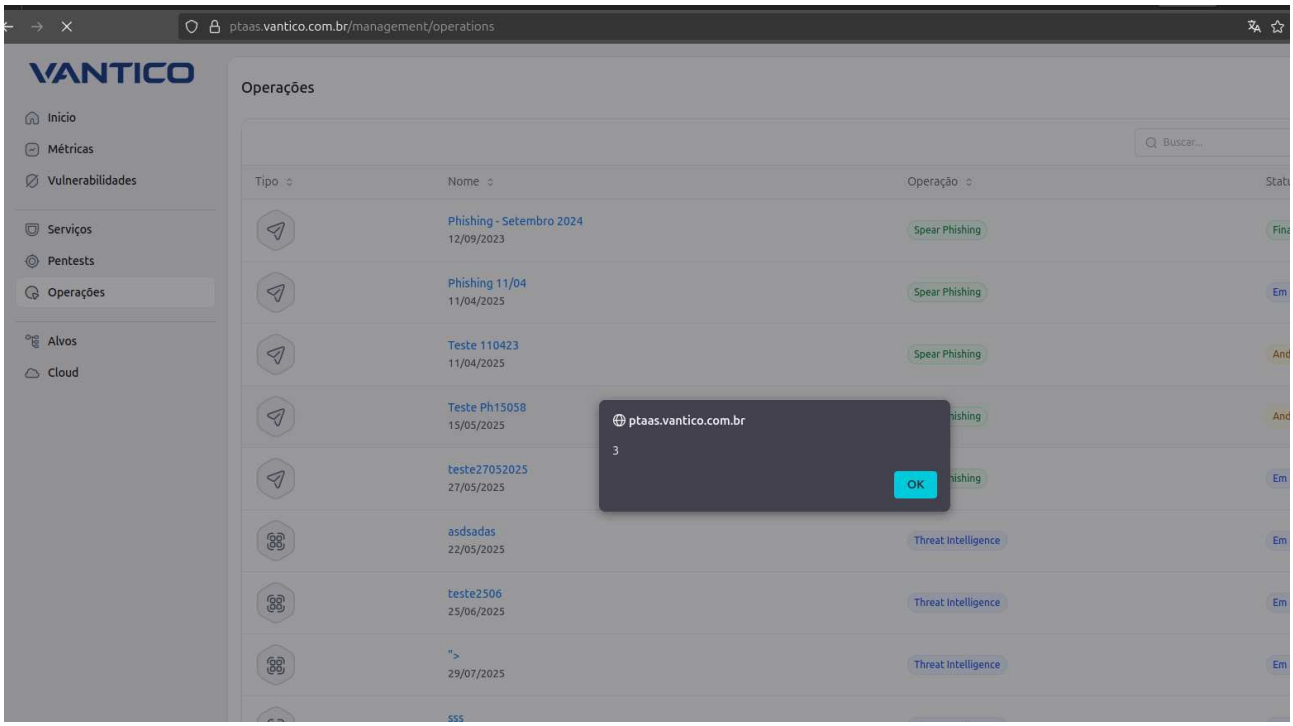

Figura 2 - Código executando na lista de operações

Esta vulnerabilidade pode trazer impactos significativos, como por exemplo:

- **Comprometimento da Confiança do Usuário:** Usuários podem ser redirecionados para sites maliciosos ou ter suas contas comprometidas.
- **Violação de Conformidade:** Regulamentações como LGPD e GDPR podem ser infringidas se informações pessoais forem comprometidas.

- **Execução de Código Arbitrário no Navegador:** Um atacante pode executar comandos JavaScript maliciosos no contexto da vítima.
- **Roubo de Sessões de Usuários:** Cookies de autenticação podem ser extraídos e reutilizados por um invasor.
- **Modificação de Conteúdo da Página:** O atacante pode alterar a interface da aplicação para induzir usuários a interagirem com elementos falsificados.
- **Exfiltração de Dados Sensíveis:** Informações pessoais e credenciais podem ser capturadas por meio de scripts maliciosos.

Recommendation

- **Sanitização e Validação de Entradas:** Implemente uma rigorosa validação e sanitização de todas as entradas de dados, especialmente aquelas provenientes dos usuários. Garanta que apenas caracteres permitidos sejam aceitos e remova ou neutralize qualquer código malicioso.
- **Utilização de Content Security Policy (CSP):** Implemente uma política de Content Security Policy (CSP) eficaz para restringir o carregamento de recursos e scripts externos. Isso pode ajudar a prevenir a execução de scripts não autorizados.
- **Escape de Dados:** Ao exibir dados no lado do cliente, utilize técnicas de escape apropriadas para garantir que os dados sejam interpretados apenas como dados, não como código executável. Isso inclui o uso de funções de escape em frameworks e bibliotecas.

5 Disclaimer

É importante observar que, embora este relatório ofereça insights valiosos sobre o estado de segurança dos serviços durante o período de teste, ele não representa a segurança geral do sistema. Alterações feitas após o teste e novos elementos introduzidos no sistema podem alterar o status de segurança. Portanto, é altamente recomendável manter uma vigilância contínua, realizar testes regulares e implementar remediações oportunas para manter uma postura de segurança robusta.