INTRODUÇÃO

As informações podem ser qualquer coisa, desde informações pessoais até perfis de mídia social, dados de telefone celular, biometria e assim por diante. O objetivo principal da segurança da informação é equilibrar a proteção da confidencialidade, integridade e disponibilidade dos dados, sem comprometer a produtividade organizacional.

Isto pode ser realizado através da implementação de um conjunto adequado de controles, incluindo políticas, regras, processos, procedimentos, estruturas organizacionais e funções de software e hardware. Para cumprir os seus objetivos específicos de segurança e de negócio, a organização deve definir, implementar, monitorizar, rever e melhorar estes controles sempre que necessário. A norma ISO/IEC 27002:2022 propõe uma abordagem sistemática para a construção de uma política de segurança sólida e eficiente.

SEGURANÇA NA WEB

A segurança na web consiste na prática de usar software, hardware, técnicas, práticas recomendadas e procedimentos de segurança para proteger websites contra ameaças externas.

Ao cumprir as diretrizes ISO 27002, as organizações podem aplicar e gerenciar seus processos de segurança, definindo de processos que visam reduzir riscos de fontes internas e de terceiros, manter a imagem da marca, assegurar os dados de clientes e proteger dados confidenciais contra vazamentos de dados.

Para essa finalidade, algumas ferramentas automatizadas de segurança, como o DAST, SAST e RASP serão exploradas nesta atividade.

- **DAST**

DAST, ou Dynamic Application Security Testing, é um método que testa o aplicativo externamente, como se um invasor estivesse tentando explorar suas vulnerabilidades. Ele simula ataques do mundo real, enviando solicitações ao aplicativo em execução e analisando suas respostas.

É importante notar que, embora o processo seja automático, essas ferramentas não estão cientes da lógica de negócio tanto quanto os atores humanos envolvidos.

Logo, as vulnerabilidades detectadas por essas ferramentas possuem desempenhos diferentes, classificados de acordo com a quantidade de alertas:

- True positive: se a ferramenta identificar um problema quando houver uma vulnerabilidade;
- False positive: se a ferramenta identificar um problema quando não há vulnerabilidades;
- True negative: se a ferramenta não identificar um problema onde não há vulnerabilidades;
- False negative: se não identificar um problema quando há uma vulnerabilidade.

Existem várias ferramentas disponíves no mercado, tais como ZAP (Zed Attack Proxy), Acunetix, Nessus, Burp Suite, AppScan etc.

- **SAST**

SAST, ou Static Application Security Testing, tem como alvo seu código fonte como fonte de detecção de vulnerabilidades. Ele examina o código linha por linha para garantir que quaisquer pontos fracos sejam revelados para que possam ser corrigidos antes do lançamento do aplicativo. Os profissionais podem testar o aplicativo de dentro para fora, com acesso à estrutura subjacente e ao design do aplicativo.

DAST e SAST são frequentemente usados em conjunto, para cobrir diferentes áreas do software. Em função da magnitude do DVWA, a revisão do código pode ser feita de forma manual (grosso modo).

Existem várias ferramentas disponíves no mercado, tais como Checkmarx, Fortify, SonarQube etc.

- **RASP**

RASP, ou Run-time Application Security Protection, atua como uma camada de proteção sobre a aplicação, diferentemente das outras ferramentas de teste utilizadas para encontrarem vulnerabilidades. Ele está conectado a um aplicativo ou ao seu ambiente e pode monitorar a execução do aplicativo; isto permite que o RASP proteja o aplicativo mesmo se as defesas do perímetro de uma rede forem violadas e os aplicativos contiverem vulnerabilidades de segurança ignoradas pela equipe de desenvolvimento.

O RASP permite que um aplicativo execute verificações de segurança contínuas em si mesmo e responda a ataques em tempo real, encerrando a sessão do invasor e alertando os defensores sobre o ataque. Opções incluem JSDefender, Veracode Runtime Protection etc.

PRINCIPAIS VULNERABILIDADES

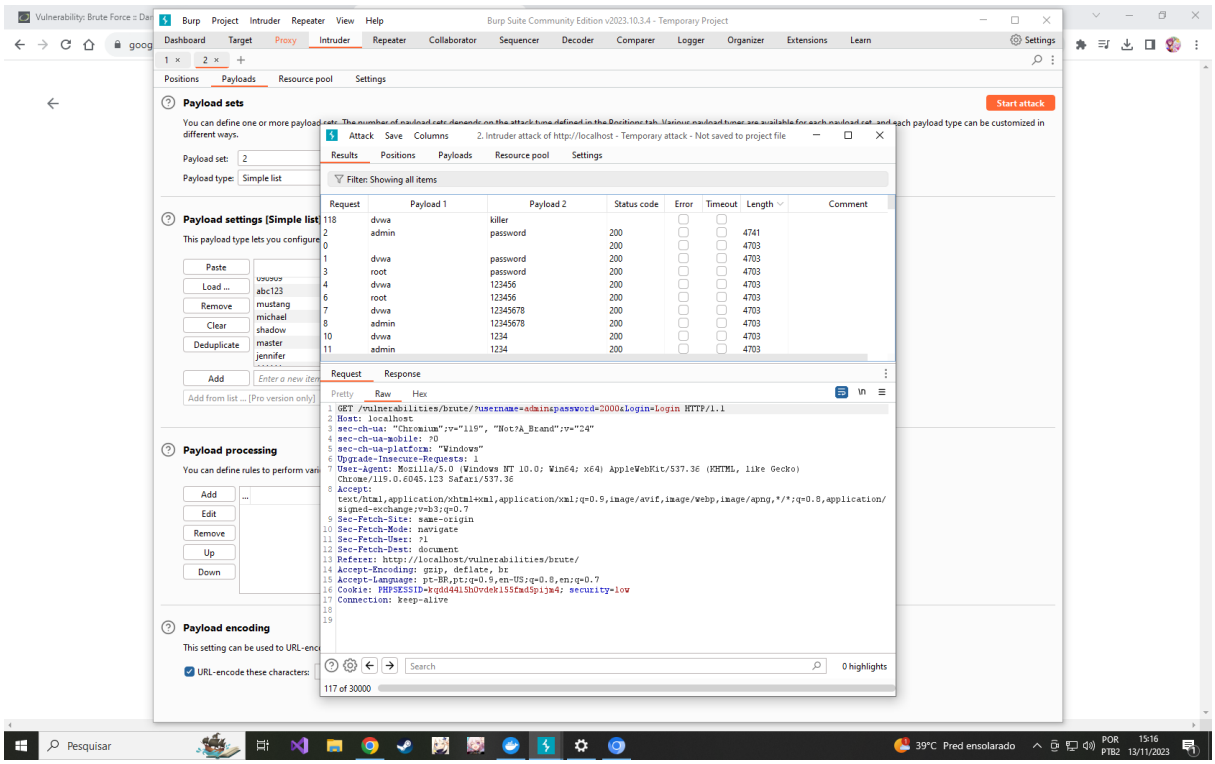
1. Brute force attacks

Um ataque de força bruta consiste em um invasor configurar valores

predeterminados, fazer solicitações a um servidor usando esses valores e, em seguida, analisar a resposta. Neste caso, o invasor verifica sistematicamente todas as senhas possíveis até que a correta seja encontrada.

Exemplo: como o aplicativo DVWA não limita a quantidade de tentativas de entrada, é possível simplesmente fazer um download de um arquivo com senhas comuns na internet e testar um por vez.

Figura 1 – Correspondência entre admin e password com Burp Suite Community (cluster attack)



Fonte: Elaboração Própria.

Formas de mitigação: utilizar CAPTCHA, bloquear endereço IP ou conta temporariamente após falhas consecutivas

2. SQL Injection

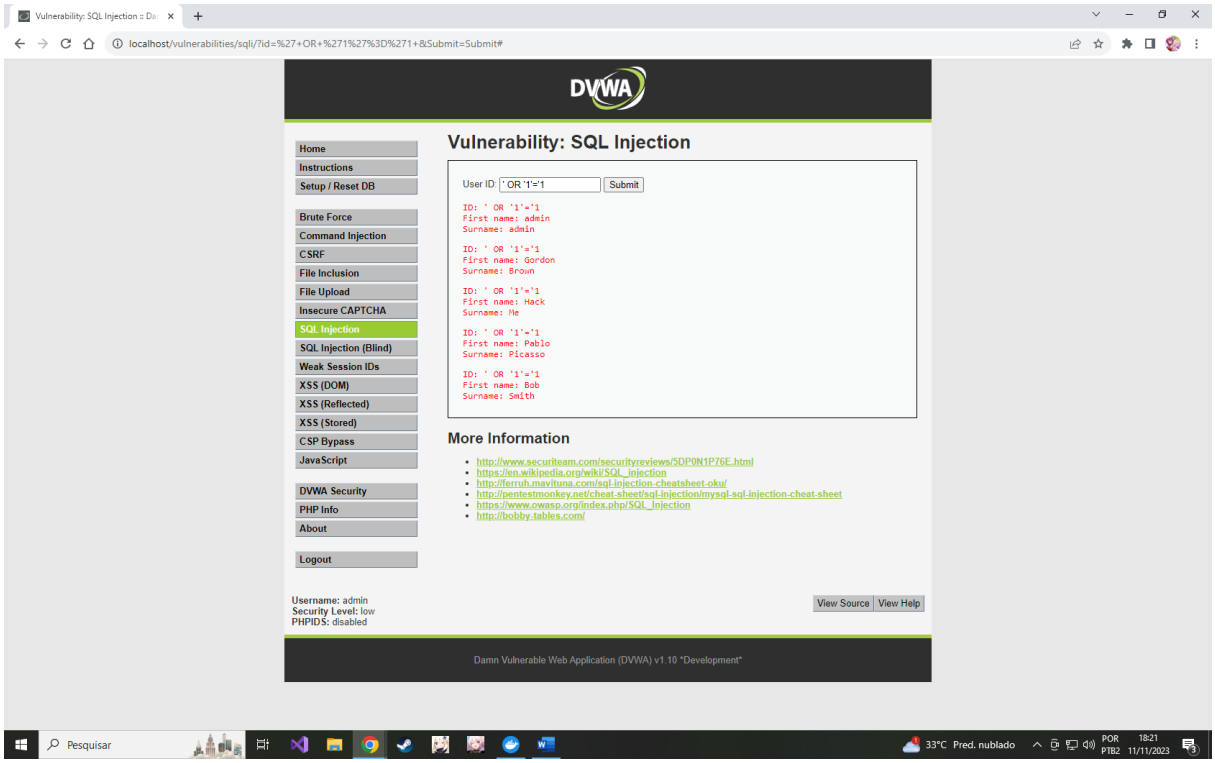
SQL Injection (SQLi) é um tipo de ataque que possibilita a execução de instruções SQL via dados de entrada. Os invasores podem contornar a autenticação e autorização de uma página da web ou aplicativo da web e recuperar o conteúdo de todo o banco de dados SQL. Eles também podem usar SQLi para adicionar, modificar e excluir registros no banco de dados.

Exemplo: A variável

```
$query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
```

irá listar todos os usuários com input ' OR '1'='1.

Figura 2 – Listando o nome de todos os usuários cadastrados



Fonte: Elaboração Própria.

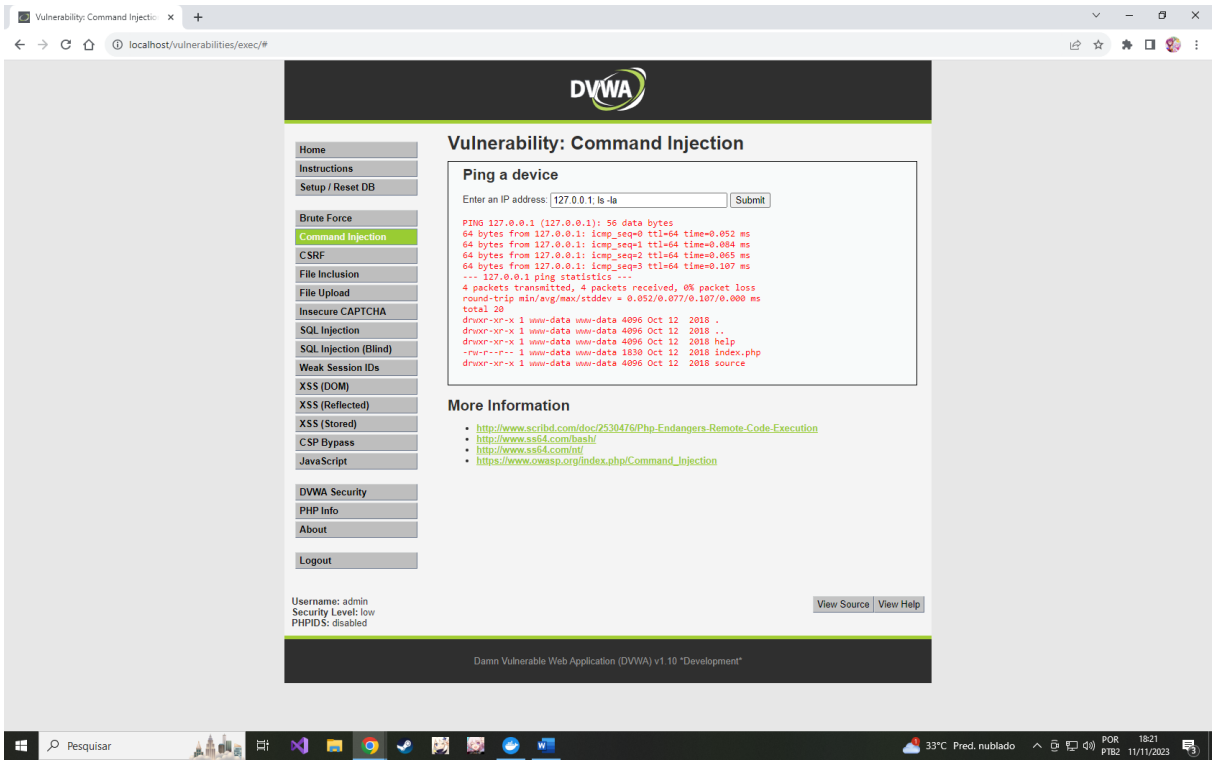
Formas de mitigação: utilizar consultas parametrizadas em vez de concatenar string diretamente, ou implementar uma validação de entrada forte fornecida pelo usuário usando métodos que defina a entrada contendo apenas caracteres alfanuméricos, por exemplo.

3. Command Injection

Ataque cujo objetivo é a execução de comandos arbitrários no sistema operacional host. Ataques de injeção de comando são possíveis quando um aplicativo passa dados inseguros fornecidos pelo usuário (formulários, cookies, cabeçalhos HTTP etc.) para um shell do sistema. Neste ataque, os comandos do sistema operacional fornecidos pelo invasor são geralmente executados com os privilégios do aplicativo vulnerável.

Exemplo: 127.0.0.1; ls -la lista os arquivos no diretório corrente.

Figura 3 - Listando todos os arquivos no diretório corrente



Fonte: Elaboração Própria.

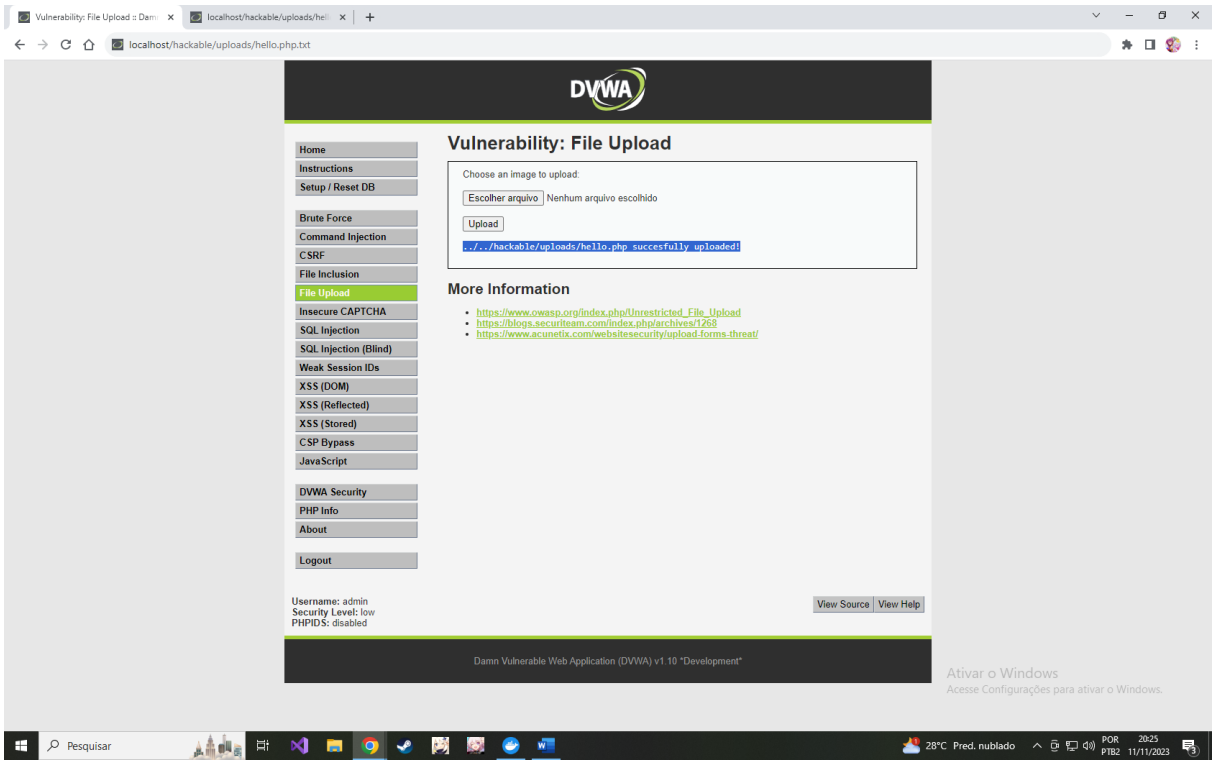
Formas de mitigação: utilizar bibliotecas de linguagem integradas, como a biblioteca “OS” do python ou a utilização de APIs, em vez de comandos do sistema operacional, além de validar a entrada fornecida pelo usuário.

4. File Upload and Path Traversal

Existem várias vulnerabilidades de inclusão de arquivos no aplicativo onde arquivos locais e externos podem ser acessados por meio de parâmetros do URL.

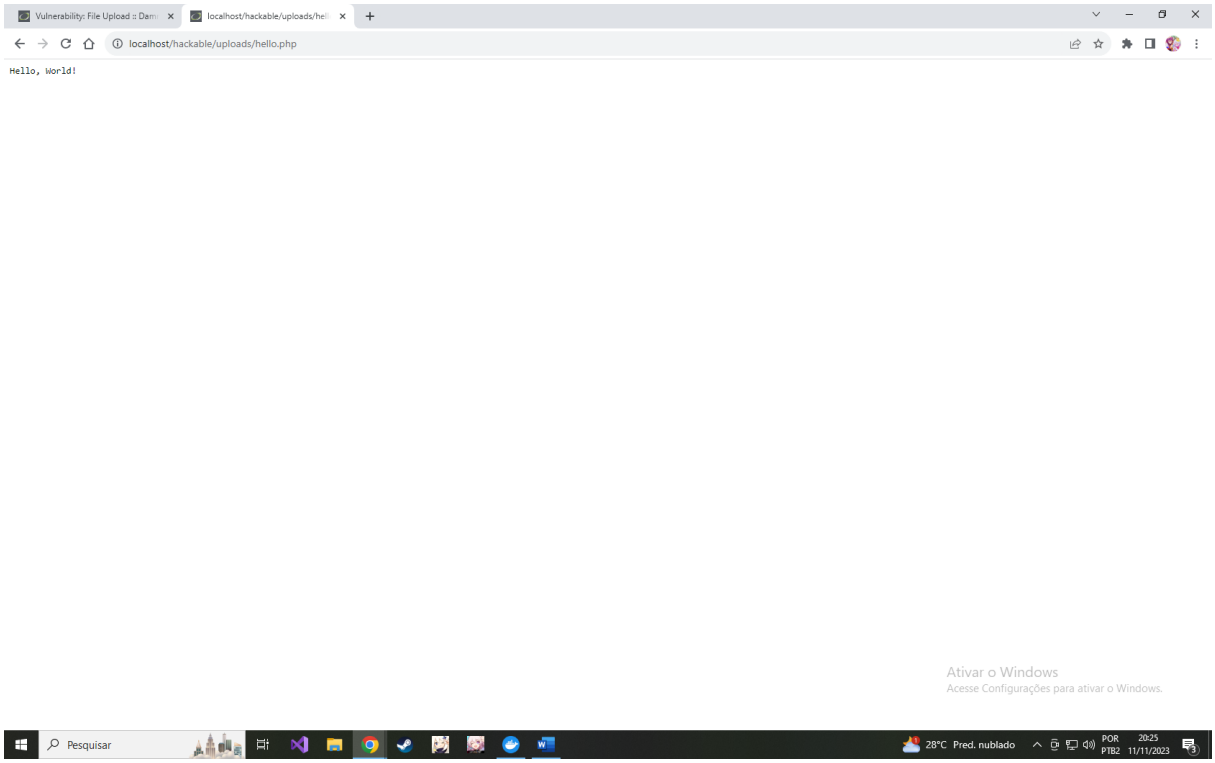
Exemplo: execução do script hello.php por meio do path relativo dado.

Figura 4 - Uploading o arquivo hello.php



Fonte: Elaboração Própria.

Figura 5 - Executando o arquivo hello.php por meio do caminho relativo



Fonte: Elaboração Própria.

Formas de mitigação: validar a entrada do usuário, restringir as permissões

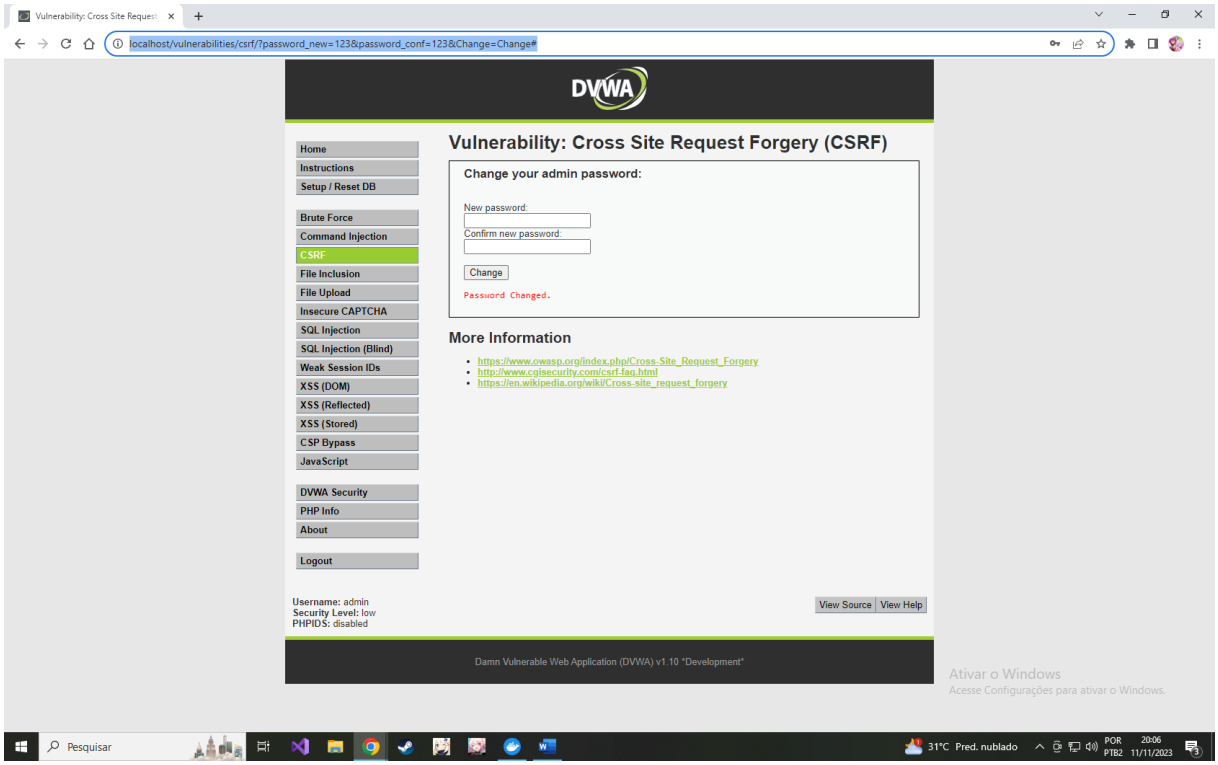
de execução, gerenciar chamadas de inclusão de arquivos (whitelist extensões de arquivos), normalizar e valide caminhos de arquivos.

5. CSRF (Cross-site Request Forgery)

Em um ataque CSRF, um usuário inocente é induzido por um invasor a enviar uma solicitação da Web que não pretendia. Isso pode causar a execução de ações no site que podem incluir vazamento inadvertido de dados de clientes ou servidores, alteração do estado da sessão ou manipulação da conta de um usuário final.

Exemplo: alterar a senha do usuário logado após clicar no seguinte link da figura 7.

Figura 6 - Trocando a senha do usuário por meio da URL



Fonte: Elaboração Própria.

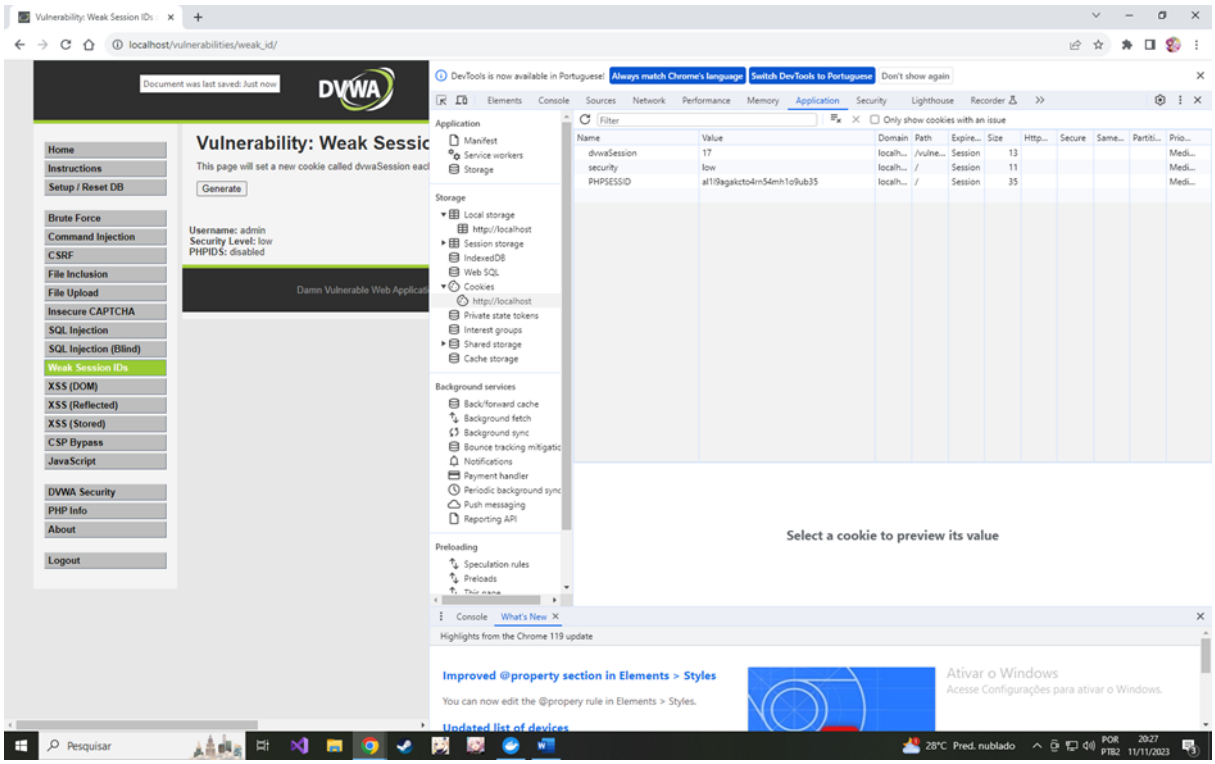
Formas de mitigação: implementar anti-CSRF tokens, SameSite cookies, implementar headers de segurança, autenticação multifator.

6. Weak Session IDs

O ataque concentra-se na previsão de valores de ID de sessão que permitem que um invasor ignore o esquema de autenticação de um aplicativo. O cookie é botão “Generate” é pressionado; com estes cookies podemos roubar uma sessão de usuário válida e nos passar por um usuário legítimo para roubar informações ou

realizar operações maliciosas.

Figura 7 - Cookie incrementando 1 por 1



Fonte: Elaboração Própria.

Formas de mitigação: utilizar o gerenciamento de sessão integrado na linguagem ou web framework, impedir a violação dos cookies (criptografar dados sensíveis, adicionar uma assinatura digital).

TENDÊNCIAS

1. **Zero Trust:** modelo de segurança cuja filosofia assume cada usuário como um ameaça potencial (dentro ou fora da rede), funcionando como uma cada extra de autenticação derivada de uma postura cética.
2. **Inteligência Artificial e Machine Learning:** a evolução da IA generativa (ChatGPT, por exemplo) traz preocupações sobre ataques automatizados e possibilidades na detecção e análise preemptiva de dados.
3. **Computação quântica:** embora esteja longe de se tornar comercialmente viável, a capacidade computacional deste sistema deve inviabilizar os métodos de criptografia convencionais.

CONCLUSÃO

Conclui-se que a proteção dos sistemas é fundamental para garantir a integridade, confidencialidade e disponibilidade dos dados.

A aplicação Damn Vulnerable Web Application (DVWA) destacou várias vulnerabilidades comuns, como ataques de força bruta, injeção de SQL e ataques CSRF. Essas vulnerabilidades ilustram a importância de abordar questões de segurança desde o início do desenvolvimento de aplicativos.

Nesse contexto, foram definidos os tipos de ferramentas e os seus papéis na identificação e prevenção de vulnerabilidades. Em resumo, a segurança na web é uma parte crítica da proteção de dados e sistemas, e a implementação de controles adequados, ferramentas de teste e boas práticas de segurança é essencial para mitigar riscos e garantir a integridade das operações do sistema.

2 REFERÊNCIAS

20 Emerging Cybersecurity Trends to Watch Out in 2024. Simplilearn, 2024. Disponível em: <https://www.simplilearn.com/top-cybersecurity-trends-article>. Acesso em: 2 mar. 2024.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO/IEC27002: Information security, cybersecurity and privacy protection — Information security controls**. Suiça, 2022.

KEN, L. **DVWA Penetration Testing Report**. Disponível em: <https://systemweakness.com/dvwa-penetration-testing-report-2c7beb3395ff>. Acesso em: 2 mar. 2024.

KOUSSA, Sherif. **What do SAST, DAST, IAST and RASP Mean to Developers?**. Software Secured, 2 nov. 2018. Disponível em: <https://www.softwaresecured.com/what-do-sast-dast-iast-and-rasp-mean-to-developers>. Acesso em: 2 mar. 2024.

OWASP. **OWASP Top Ten**. Disponível em: <https://owasp.org/www-project-top-ten>. Acesso em: 2 mar. 2024.

WOOD, R. **Damn Vulnerable Web Application**. Disponível em: <https://github.com/digininja/DVWA>. Acesso em: 2 mar. 2024.