CESAR School
Segurança da Informação
Prof.: Henrique Arcoverde
Aluno: Bernardo Queiroz

Relatório
de
Avaliação de Segurança
Physecure

Recife - PE
2023

RESUMO

Este relatório descreve o resultado dos testes de segurança realizados na aplicação Physecure. Foram simuladas situações de ataques realizados por um ator externo com os objetivos de ganhar acesso ao sistema, coletar informações sensíveis, deixar o serviço fora de operação e realizar ações indesejadas. Foram identificadas um total de 16 vulnerabilidades que podem ser classificadas nos graus de risco: Crítico, Alto, Médio, Baixo e Info.

Foram encontradas 10 vulnerabilidades que permitem ações de risco crítico ou alto para aplicação, permitindo que um ator malicioso realize ações como por exemplo, acessar e enviar qualquer arquivo para o servidor, acessar todo o banco de dados, logar na conta de qualquer usuário e deixar a aplicação fora de operação. Diante de tal situação, sugerimos uma completa refatoração do sistema e para cada vulnerabilidade encontrada deixaremos uma recomendação para solução.

DEFINIÇÃO DAS CLASSIFICAÇÕES

Classificação dos riscos

Nível	Score	Descrição
Crítico	**10**	A vulnerabilidade representa uma ameaça imediata para a organização. A exploração bem-sucedida pode afetar permanentemente a organização. A correção deve ser realizada imediatamente.
Alto	**7-9**	A vulnerabilidade representa uma ameaça urgente para a organização e a correção deve ser priorizada.
Médio	**4-6**	A exploração bem-sucedida é possível e pode resultar em interrupção notável da funcionalidade do negócio. Essa vulnerabilidade deve ser corrigida quando possível.
Baixo	**1-3**	A vulnerabilidade representa uma ameaça insignificante/mínima para a organização. A presença dessa vulnerabilidade deve ser observada e corrigida, se possível.
Info	**0**	Essas descobertas não representam uma ameaça clara para a organização, mas podem fazer com que os processos de negócios funcionem de maneira diferente do desejado ou revelar informações confidenciais sobre a empresa.

Classificação das probabilidades de exploração

Probabilidade	Descrição
Provável	Os métodos de exploração são bem conhecidos e podem ser executados usando ferramentas disponíveis publicamente. Atacantes pouco qualificados e ferramentas automatizadas podem explorar com sucesso a vulnerabilidade com dificuldade mínima.
Possível	Os métodos de exploração são bem conhecidos, podem ser executados usando ferramentas públicas, mas requerem configuração. A compreensão do sistema subjacente é necessária para uma exploração bem-sucedida.

Improvável	A exploração requer uma compreensão profunda dos sistemas subjacentes ou habilidades técnicas avançadas. Condições precisas podem ser necessárias para uma exploração bem-sucedida.

Classificação do impacto ao negócio

Impacto	Descrição
Severo	A exploração bem-sucedida pode resultar em grandes interrupções de funções críticas de negócios em toda a organização e danos financeiros significativos.
Moderado	A exploração bem-sucedida pode causar interrupções significativas em funções de negócios não críticas.
Menor	A exploração bem-sucedida pode afetar poucos usuários, sem causar muita interrupção nas funções de negócios de rotina.

Classificação de dificuldade de remediação

Dificuldade	Descrição
Difícil	A correção pode exigir uma extensa reconfiguração dos sistemas subjacentes que consome muito tempo. A correção pode exigir a interrupção das funções comerciais normais.
Médio	A correção pode exigir pequenas reconfigurações ou adições que podem ser demoradas ou caras.
Fácil	A correção pode ser realizada em um curto espaço de tempo, com pouca dificuldade.

RESULTADOS DA AVALIAÇÃO

Número	Vulnerabilidade	Score	Risco	Página
1	Upload de script	**10**	**Crítico**	6
2	Exposição do arquivo backup.sql	**10**	**Crítico**	8
3	Uso de http em toda a aplicação	**10**	**Crítico**	9
4	Bypass no login com SQL-Injection	**9**	**Alto**	10
5	Falha na geração do token de sessão	**9**	**Alto**	11
6	Senhas salvas em texto plano	**8**	**Alto**	12
7	Cross-site scripting persistido	**8**	**Alto**	13
8	Cross-site scripting refletido	**7**	**Alto**	14
9	IDOR na URL de atendimento	**7**	**Alto**	16
10	CSRF	**7**	**Alto**	17
11	Falha no mecanismo de proteção contra força bruta	**6**	**Médio**	18
12	Enumeração de usuários	**6**	**Médio**	19
13	Token de sessão com atributo HTTPOnly com valor false	**6**	**Médio**	20
14	Política de senhas extremamente fraca	**6**	**Médio**	22
15	Indicação da política de matrícula	**3**	**Baixo**	23
16	Ausência de segundo fator de autenticação	**0**	**Info**	24

1 - Upload de Script

RISCO CRÍTICO (10/10)	
Probabilidade	**Provável**
Impacto	**Severo**
Remediação	**Fácil**

Implicações de Segurança

Essa vulnerabilidade é uma das mais graves que podem acontecer, a possibilidade de upload de um script implica num total controle do sistema sendo viável executar qualquer tipo de ação maliciosa do lado do servidor.

Análise

O mecanismo de troca de foto de perfil está completamente vulnerável ao upload de scripts maliciosos, não é necessário um alto nível de conhecimento para encontrar e explorar esse bug. Como prova de conceito, nós enviamos um arquivo .php que consegue executar qualquer comando no terminal do servidor, sendo possível desligar a máquina, acessar informações e apagar arquivos.

Para reproduzir esse exploit específico contamos com o auxílio do arquivo *text.php*, acessível pelo path "*/purple/text.php*", com ele é possível carregar e executar qualquer arquivo .php. Em seguida, basta criar um script malicioso que receba os comandos por algum dos parâmetros da URL, no nosso caso chamamos esse parâmetro de "*command*", depois é necessário acessar a página para mudança de foto de perfil, adicionar o script para o envio, clicar no botão de enviar, interceptar a requisição pelo proxy, alterar o path de upload para "*/tmp/*" e, por fim, acessar o path "*/purple/text.php?file=/tmp/<nome do arquivo malicioso>.php&command=<comando para executar no terminal>*". Fazendo isso o arquivo *text.php* irá executar o arquivo malicioso passando o comando descrito no parâmetro "*command*" da URL e exibir o resultado no html da resposta.

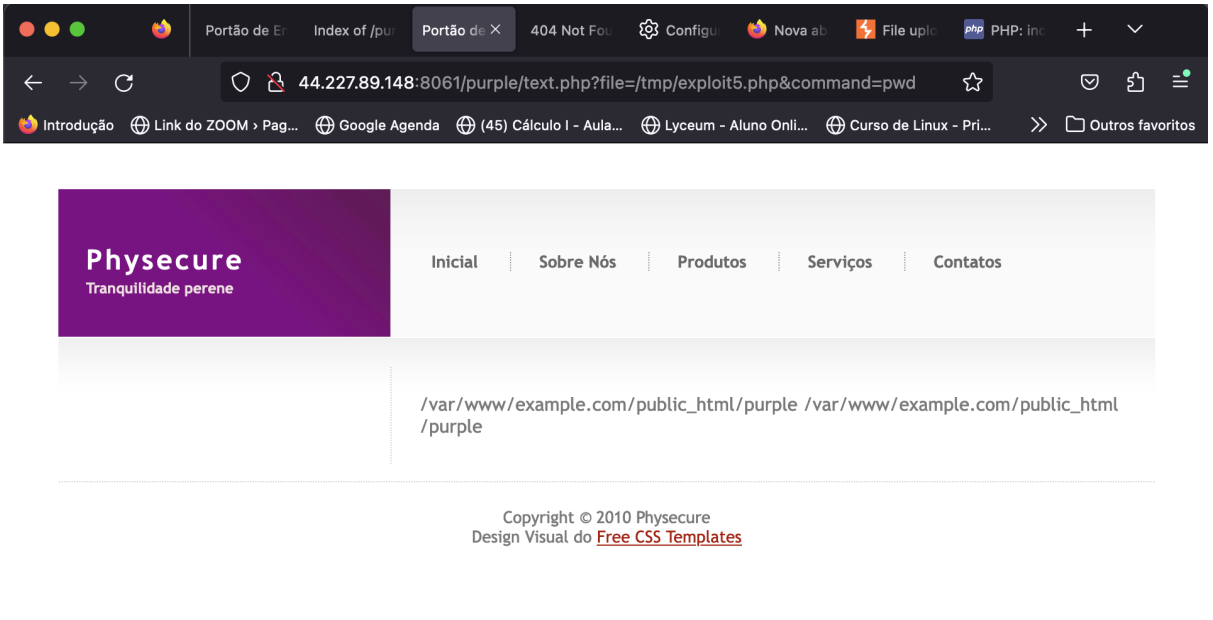

***Imagem 1**: execução do comando pwd no terminal do servidor*

Recomendações para solução

- Especificar de forma restrita quais são as extensões de arquivos permitidas para upload
- Renomear os arquivos recebidos para evitar sobrescrita de arquivos existentes que possuem mesmo nome e extensão do arquivo enviado
- Não salvar arquivos diretamente no filesystem do servidor até que eles sejam propriamente avaliados
- Uso de frameworks para auxiliar no desenvolvimento de features para upload de arquivos

Referências

- https://portswigger.net/web-security/file-upload

2 - Exposição do arquivo backup.sql

RISCO CRÍTICO (10/10)	
Probabilidade	**Provável**
Impacto	**Severo**
Remediação	**Fácil**

Implicações de Segurança

Essa vulnerabilidade é extremamente crítica pois é possível acessar todo o banco de dados da aplicação, expondo as informações registradas nele. Tal fato impacta bastante na credibilidade da aplicação e na segurança das informações dos usuários e da aplicação como um todo.

Análise

É possível acessar o arquivo *backup.sql* diretamente pelo path: "*/purple/backup.sql*". Isso fere o princípio da redução da superfície de ataque, pois não há necessidade desse arquivo está exposto dessa forma na internet, arquivos com dados sensíveis devem exigir alguma forma de autenticação para serem acessados e preferencialmente estar numa rede privada.

Recomendações para solução

- Salvar os arquivos relativos ao banco de dados num servidor a parte do resto da aplicação
- Não expor arquivos sensíveis para a internet sem exigir formas seguras de autenticação

Referências

- https://www.fortinet.com/resources/cyberglossary/attack-surface#:~:text=The%20smaller%20the%20attack%20surface,the%20risk%20of%20cyberattacks%20succeeding.

3 - Uso do http em toda a aplicação

RISCO CRÍTICO (10/10)	
Probabilidade	**Provável**
Impacto	**Severo**
Remediação	**Médio**

Implicações de Segurança

Toda a aplicação está se comunicando através do protocolo http, ele não oferece uma camada de criptografia das mensagens, tornando a aplicação extremamente vulnerável a ataques de interceptação das mensagens. Nesse caso será possível ver o payload das mensagens e, consequentemente, encontrar dados sensíveis, como as credenciais de login dos usuários.

Análise

Não se deve utilizar o http para trafegar informações com segurança, como o protocolo http não oferece criptografia para as mensagens e o canal de comunicação, todo o payload das requisições e respostas podem ser lidos sem dificuldade por terceiros mal intencionados. Na tela de login e nas seguintes a aplicação deve utilizar exclusivamente o protocolo https.

Recomendações para solução

- Na tela de login e nas telas após autenticação utilizar obrigatoriamente o protocolo https
- Utilizar o http apenas em telas que não exigem autenticação com exceção da tela de login

Referências

- https://www.fortinet.com/resources/cyberglossary/attack-surface#:~:text=The%20smaller%20the%20attack%20surface,the%20risk%20of%20cyberattacks%20succeeding.

4 - Bypass no login com SQL-Injection

RISCO ALTO (9/10)	
Probabilidade	**Provável**
Impacto	**Severo**
Remediação	**Fácil**

Implicações de Segurança

A aplicação está vulnerável a sql injection nos campos de login, com isso é possível logar na conta de todos os usuários.

Análise

Esse tipo de bug permite ao atacante alterar diretamente as queries que estão rodando na aplicação. Nesse caso em específico a requisição POST de login foi modificada para no body a matrícula vir com o payload: *<número da matrícula>'--+*, dessa forma a senha será ignorada na query, pois virá após a notação de comentário, e o login será realizado.

***Imagem 2**: requisição POST alterada para burlar login com sql-injection*

Recomendações para solução

- Uso de prepared statements para montar as queries no backend (queries pré-computadas)

Referências

- https://portswigger.net/web-security/sql-injection

5 - Falha na geração do token de sessão

RISCO ALTO (9/10)	
Probabilidade	**Provável**
Impacto	**Severo**
Remediação	**Fácil**

Implicações de Segurança

O token de sessão possui falha na sua geração pois é facilmente previsível, isso torna extremamente fácil ataques de sequestro de sessão.

Análise

Foi utilizado para gerar os tokens de sessão um mecanismo muito vulnerável, basicamente os tokens são gerados a partir do encoder do número da matrícula em ASCII Hex. Isso é uma estratégia bem conhecida pelos hackers e não é necessário grande nível técnico para explorar essa vulnerabilidade.

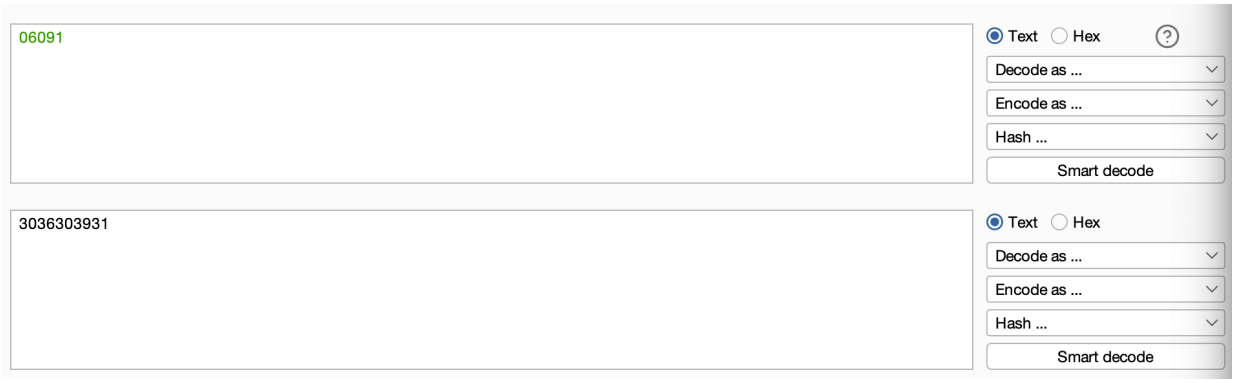

***Imagem 3**: realizando encoder da matrícula para descobrir o token de sessão*

Recomendações para solução

- Utilizar como token um valor aleatório e diferente para cada sessão.

Referências

- https://cqr.company/web-vulnerabilities/insecure-token-generation/

6 - Senhas salvas em texto plano

RISCO ALTO (8/10)	
Probabilidade	**Possível**
Impacto	**Severo**
Remediação	**Médio**

Implicações de Segurança

Ao acessar o banco de dados foi visto que as senhas de login dos usuários são salvas em texto plano. Qualquer pessoa que tenha acesso ao banco pode ver essas informações, isso é bastante grave pois provavelmente alguns usuários devem usar a mesma senha para mais de uma aplicação, levando a um potencial de prejuízo muito maior para eles.

Análise

As senhas dos usuários nunca devem ser salvas no banco em texto plano, pois dessa forma estarão vulneráveis tanto à hackers que consigam burlar os mecanismos de controle de acesso ao banco quanto aos funcionários da Physecure que possuem esse acesso.

Recomendações para solução

- Utilizar um hash criptográfico, como o MD5, ou um algoritmo de criptografia simétrica, como o AES, para salvar e validar as senhas.
- Usar um salt antes da geração do hash ou da criptografia, isso é uma forma de evitar o uso de rainbow tables no caso do hash criptográfico e complicar a vida do atacante caso ele encontre a chave do algoritmo de criptografia

Referências

- https://www.alura.com.br/artigos/como-armazenar-senhas-no-banco-de-dados-de-forma-segura

7 - Cross-site scripting persistido

RISCO ALTO (8/10)	
Probabilidade	**Provável**
Impacto	**Severo**
Remediação	**Fácil**

Implicações de Segurança

Esse tipo de ataque permite ao atacante injetar código javascript malicioso de forma persistente na aplicação. Dessa forma é possível realizar uma grande variedade de ações indesejadas pelo usuário, como por exemplo, alterar o status de um incidente ou mudar a senha do usuário.

Análise

A tela de atendimento possui um campo de observações onde é possível injetar javascript de forma persistente. Esse ataque é potencialmente mais perigoso que o cross-site scripting refletido, pois para ser realizado não exige uma ação fora do padrão realizada pelo usuário. Como o script estará persistindo na aplicação, quando o usuário logar na página principal o ataque será realizado automaticamente.

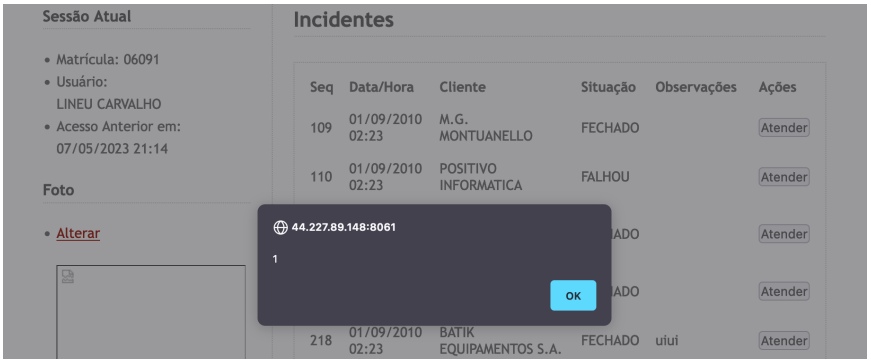

***Imagem 4**: prova de conceito, sempre que o usuário abrir a página principal esse alerta será exibido*

Recomendações para solução

- Filtrar o input de forma que seja aceito apenas os caracteres esperados no campo específico
- Realizar o encode do output, dessa forma as entidades html não serão interpretadas como código

Referências

- https://portswigger.net/web-security/cross-site-scripting

8 - Cross-site scripting refletido

RISCO ALTO (7/10)	
Probabilidade	**Provável**
Impacto	**Severo**
Remediação	**Fácil**

Implicações de Segurança

Esse tipo de ataque permite ao atacante injetar código javascript malicioso de forma refletida na aplicação. Dessa forma é possível realizar uma grande variedade de ações indesejadas pelo usuário, como por exemplo, alterar o status de um incidente ou mudar a senha do usuário.

Análise

No POST de login é possível injetar javascript de forma refletida no campo de matrícula e no campo "*seq*" da URL para abrir um atendimento também é possível realizar ataques de XSS. Esse ataque é potencialmente menos perigoso que o cross-site scripting persistido, pois exige que o usuário acesse uma página maliciosa ou clique num link malicioso, logo para realizar esse tipo de ataque também é necessário um pouco de engenharia social.

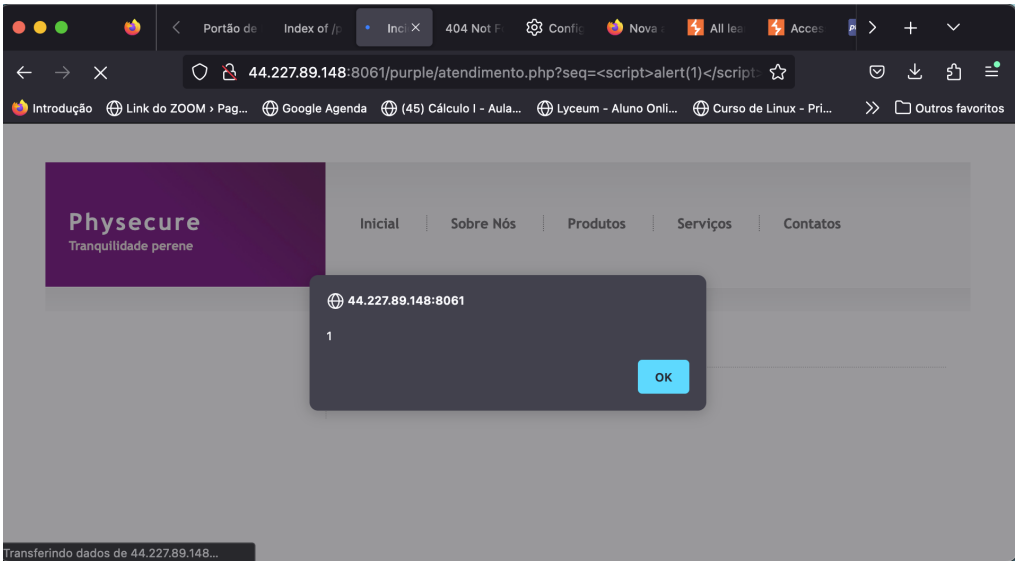

***Imagem 5**: prova de conceito, passando o valor <script>alert(1)</script> no parâmetro "seq" da URL é possível executar esse alert(1)*

Recomendações para solução

- Realizar o encode do output, dessa forma as entidades html não serão interpretadas como código

Referências

- https://portswigger.net/web-security/cross-site-scripting

9 - IDOR na URL de atendimento

RISCO ALTO (7/10)	
Probabilidade	**Provável**
Impacto	**Severo**
Remediação	**Fácil**

Implicações de Segurança

Esse tipo de ataque permite ao hacker acessar recursos da aplicação que são restritos para outros usuários, logo é uma forma de burlar o controle de acesso.

Análise

Nesse caso em específico é possível acessar a página de todos os atendimentos, basta alterar o valor do parâmetro "*seq*". Não é necessário sequer estar logado em uma sessão autenticada, basta enviar a URL no formato correto.

Recomendações para solução

- Realizar verificação do token de sessão para garantir um controle de acesso adequado para o recurso

Referências

- https://portswigger.net/web-security/access-control/idor

10 - CSRF

RISCO ALTO (7/10)	
Probabilidade	**Possível**
Impacto	**Severo**
Remediação	**Médio**

Implicações de Segurança

Esse tipo de ataque permite ao hacker forgar requisições que se passam como feitas por um usuário legítimo e termina realizando ações indesejadas. Na Physecure, por exemplo, é possível alterar a senha dos usuários.

Análise

Para validar essa vulnerabilidade foi criado um arquivo html que poderia ser um website malicioso, ao abrir esse html automaticamente é realizado um POST para troca de senha na Physecure. Como o cookie de sessão possui o atributo same site com o valor none, o token termina sendo enviado na requisição e a troca de senha é realizada com sucesso.

Recomendações para solução

- Colocar o atributo de same-site para strict no cookie de sessão. Dessa forma, requisições forjadas não vão enviar o token de sessão e consequentemente, caso o mecanismo de sessão esteja funcionando corretamente, a troca de senha não será autorizada
- Uso de CSRF tokens, esse tipo de token é gerado pelo servidor e fornecido para o cliente numa página html pré-envio da requisição. Como para ter acesso a esse token é preciso abrir a página onde o token é fornecido, a dificuldade para realizar ataques CSRF aumenta muito.

Referências

- https://portswigger.net/web-security/csrf

11 - Falha no mecanismo de proteção contra força bruta

RISCO MÉDIO (6/10)	
Probabilidade	**Provável**
Impacto	**Moderado**
Remediação	**Fácil**

Implicações de Segurança

Falhas nesse mecanismo permitem a realização de ataques de força bruta com as mais variadas intenções.

Análise

No caso da Physecure estamos falando do cookie "FALHAS", este cookie tem como função contar a quantidade de tentativas de login falhadas para que a partir da quinta requisição o mecanismo de login fique indisponível para o cliente. Porém, como todo mecanismo no lado do cliente, está sujeito a manipulação, logo é possível alterar o valor desse cookie e consequentemente burlar o mecanismo de proteção contra força bruta no login.

Recomendações para solução

- Utilizar mecanismos contra força bruta que atuem no lado do servidor, como por exemplo, bloquear login com base no número de tentativas falhas por matrícula.
- Uso de CAPTCHA

Referências

- https://www.cloudways.com/blog/what-is-brute-force-attack/

12 - Enumeração de usuários

RISCO MÉDIO (6/10)	
Probabilidade	**Provável**
Impacto	**Moderado**
Remediação	**Fácil**

Implicações de Segurança

Tal vulnerabilidade permite ao atacante obter todos os usuários existentes na aplicação.

Análise

A resposta de login presente no html indica a existência do usuário em questão, quando o número de matrícula não corresponde a um usuário existente a mensagem é "<núm de matrícula fornecido> não encontrado…", já quando o usuário existe e a senha está errada a mensagem é "senha incorreta…". Com isso é possível realizar um ataque de força bruta tentando todas as permutações de matrícula possíveis e filtrar os usuários válidos pela resposta presente no html.

Recomendações para solução

- Fornecer uma mensagem genérica no html de resposta do login, uma que não indique se é o usuário ou a senha que está incorreto. Ex.: "Usuário ou senha incorretos, tente novamente"

Referências

- https://portswigger.net/blog/preventing-username-enumeration

13 - Token de sessão com atributo HTTPOnly com valor false

RISCO MÉDIO (6/10)	
Probabilidade	**Possível**
Impacto	**Moderado**
Remediação	**Fácil**

Implicações de Segurança

Esse tipo de configuração permite uma maior exploração dos ataques de cross-site scripting, pois será possível manipular o cookie de sessão através do script realizado, essa é uma porta de entrada clara para ataques de sequestro de sessão.

Análise

Quando o cookie de sessão não possui o atributo HTTPOnly configurado como true, ele será passível de manipulação nos ataques de cross-site scripting. Dessa forma é possível acessá-lo através do objeto *document* em seu atributo *cookies (document.cookie).*

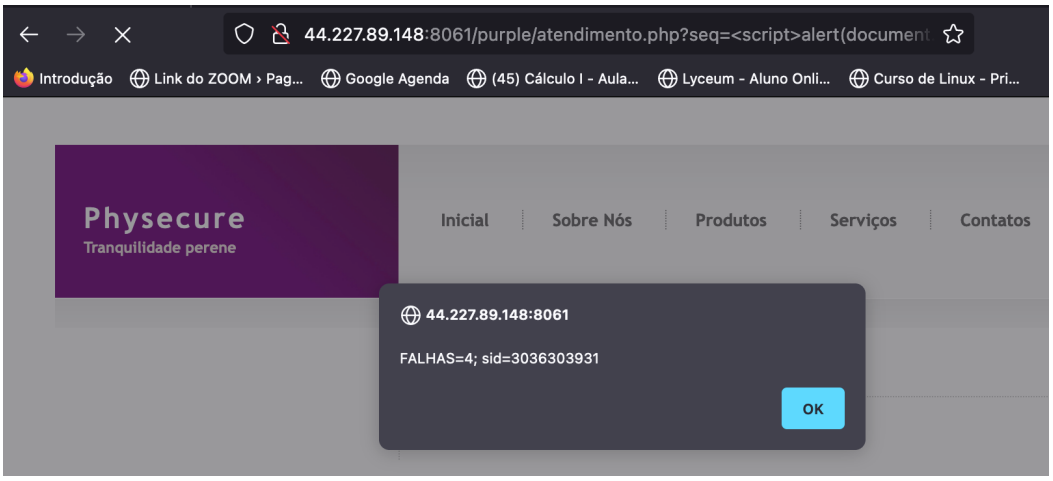

***Imagem 5**: prova de conceito, passando o valor <script>alert(document.cookie)</script> no parâmetro "seq" da URL é possível exibir os cookies do usuário*

Recomendações para solução

- Ao enviar o cookie de sessão para o cliente, passar junto a ele o atributo HTTPOnly com valor *true*

Referências

- https://owasp.org/www-community/HttpOnly

14 - Política de senhas extremamente fraca

RISCO MÉDIO (6/10)	
Probabilidade	**Provável**
Impacto	**Severo**
Remediação	**Média**

Implicações de Segurança

Tal vulnerabilidade permite que os usuários atribuam senhas muito fracas nas suas respectivas contas, isso torna bem mais simples os ataques de força bruta na senha.

Análise

A política da Physecure permite até senhas de apenas um dígito, isso é uma realidade bem longe da segurança, pois muitos usuários não vão ter conhecimento do que é uma senha segura e provavelmente escolherão senhas bastante vulneráveis a ataques de força bruta.

Recomendações para solução

- Exigir senhas de pelo menos 8 caracteres
- Exigir presença de caracteres especiais
- Exigir presença de letras maiusculas e minusculas
- Exigir presença de números
- Lembrando que o rigor da política de senhas deve ser compatível com o nível de sensibilidade da aplicação.

Referências

- https://en.wikipedia.org/wiki/Password_policy

15 - Indicação da politica de matrícula

RISCO BAIXO (3/10)	
Probabilidade	**Provável**
Impacto	**Menor**
Remediação	**Fácil**

Implicações de Segurança

Permite que os atacantes tomem conhecimento do formato exato que a matrícula exige, isso facilita os ataques de enumeração de usuários.

Análise

Na página de login, antes de enviar para o servidor a requisição POST, é feita uma validação do formato da matrícula via regex. Isso permite descobrir facilmente que a matrícula deve ser composta apenas de números e possuir até cinco dígitos, essa informação define o escopo exato de um ataque de força bruta para enumeração dos usuários da aplicação.

Recomendações para solução

- Não realizar validação de matrícula no frontend, essa validação deve ser feita no backend e não deve fornecer indicativos para o cliente

16 - Ausência de segundo fator de autenticação

RISCO INFO (0/10)	
Probabilidade	**Provável**
Impacto	**Severo**
Remediação	**Difícil**

Implicações de Segurança

A ausência de um segundo fator de autenticação é algo que torna o mecanismo de autenticação mais vulnerável.

Análise

Isso se caracteriza mais como uma recomendação do que uma vulnerabilidade, pois não é algo que seja obrigatório para todas as aplicações, o uso de um 2FA é uma decisão que deve levar em conta o nível de segurança que a aplicação exige, assim como a política de senhas.

Recomendações para solução

- Utilizar algum framework que auxilia na implementação do mecanismo de segundo fator de autenticação.

Referências

- https://experience.dropbox.com/pt-br/resources/what-is-2fa