FACULDADE DE TECNOLOGIA E CIÊNCIAS SOCIAIS APLICADAS – FATECS
ENGENHARIA DE COMPUTAÇÃO

NICHOLAS FIALHO CABRAL
22153184

THIAGO GOMES SILVA
22153205

TESTE DE PENETRAÇÃO: UTILIZAÇÃO DA METODOLOGIA OWASP EM APLICAÇÕES WEB.

BRASÍLIA
2022

NICHOLAS FIALHO CABRAL
THIAGO GOMES SILVA

TESTE DE PENETRAÇÃO: UTILIZAÇÃO DA METODOLOGIA OWASP EM APLICAÇÕES WEB.

Trabalho de Conclusão de Curso (TCC) apresentado como um dos requisitos para a conclusão do curso de Engenharia de Computação do UniCEUB – Centro Universitário de Brasília

Orientador (a)**: Prof. MsC Francisco Javier de Obaldia Diaz**

BRASÍLIA
2022

NICHOLAS FIALHO CABRAL
THIAGO GOMES SILVA

TESTE DE PENETRAÇÃO: UTILIZAÇÃO DA METODOLOGIA OWASP EM APLICAÇÕES WEB.

Trabalho de Conclusão de Curso (TCC) apresentado como um dos requisitos para a conclusão do curso de Engenharia Computação do UniCEUB – Centro Universitário de Brasília

Orientador (a)**: Prof. MsC Francisco Javier de Obaldia Diaz**

Brasília, 2022.

BANCA EXAMINADORA

_____
Prof. MsC Francisco Javier de Obaldia Diaz
Orientador (a)

_____
Prof. Fabio Oliveira Guimaraes
Examinador

_____
Prof. MsC William Roberto Malvezzi
Examinador

TESTE DE PENETRAÇÃO: UTILIZAÇÃO DA METODOLOGIA OWASP EM APLICAÇÕES WEB.

PENTEST : USING THE OWASP METHODOLOGY IN WEB APPLICATIONS.

Nicholas Fialho Cabral[1], Thiago Gomes da Silva[1], Francisco Javier de Obaldia Diaz[2], Fabio Oliveira Guimaraes[3], William Roberto Malvezzi[4]

RESUMO

Com a crescente utilização e oferta de serviços da internet, observa-se que uma nova área de exploração foi formada: a de proteção de dados pessoais expostos na web, que, de forma massiva, essa exposição vem sendo combatida no presente com a criação de softwares cada vez mais seguros. Dentro desse ponto, o OWASP (Projeto Aberto de Segurança em Aplicações Web), uma comunidade reconhecida por suas metodologias e documentações sobre os mais diversos tipos de tratamentos de vulnerabilidades em ambiente web, apresenta a cada quatro anos uma lista com as dez vulnerabilidades mais relevantes em questões de ataques hackers. Neste trabalho será feita uma descrição objetiva de cada uma dessas vulnerabilidades, apresentando a seguinte metodologia: reconhecimento, enumeração de vulnerabilidades, exploração das vulnerabilidades, pós-exploração e apresentação. Este artigo tem como objetivo apresentar a metodologia utilizada em um ambiente web, demonstrando como cada etapa foi executada, as vulnerabilidades encontradas e a apresentação destas para mitigação em um ambiente real.

Palavras-chave: Segurança. Pentest. OWASP. Metodologia. Web.

Abstract:

With the increasing use and supply of internet services, it is observed that a new area of exploration has been formed: the protection of personal data exposed on the web, which, in a massive way, this exposure has been fought at present with the creation of increasingly secure software. Within this point, the OWASP (Open Web Application Security Project), a community recognized for its methodologies and documentation on the most diverse types of vulnerability treatment in the web environment, presents a list of the ten most relevant vulnerabilities every four years. in matters of hacker attacks. In this work, an objective description of each of these vulnerabilities will be made, presenting the following methodology: recognition, enumeration of vulnerabilities, exploitation of vulnerabilities, post-exploitation and presentation. This article aims to present the methodology used in a web environment, demonstrating how each step was performed, the vulnerabilities found and their presentation for mitigation in a real environment.

keywords: Security. Pentest. OWASP. Methodology. Web.

[1] UniCEUB, aluno. [2] UniCEUB, orientador. 3 UniCEUB, primeiro examinador. [4] UniCEUB, segundo examinador.

1 INTRODUÇÃO

Nas duas últimas décadas, é possível observar a crescente evolução da tecnologia na vida das pessoas, estando presente desde o entretenimento até o dia a dia profissional. Com esse avanço tecnológico a população é atingida por uma inundação de dados e informações, onde é realizada a troca de informações com a rede mundial de computadores, desde o nome completo até dados do cartão de crédito, é possível perder de vista a rede social e o marketplace no qual os dados pessoais são cadastrados.

Seguindo o fluxo dos internautas, as empresas que além de suas informações, recebem inúmeros dados pessoais de clientes, conteúdo sensível, que hoje sua privacidade é regida pela Lei Geral de Proteção de Dados Pessoais (LGPD), que estabelece as condições nas quais devem ser tratados, desde seu armazenamento até o compartilhamento com terceiros e sendo passível de sanções administrativas caso não seja cumprido.

Tendo esse cenário como palco, as empresas começam a enxergar a importância da segurança da informação, começando a investir em uma equipe capacitada e em ferramentas fundamentais para realizar essa gestão, aliada a essas ações é importante utilizar uma metodologia atual e adequada, tendo em vista o aumento da complexidade das aplicações e a facilidade da disseminação do conhecimento que possa ser utilizado para ataques e roubos de informações, como relata STALLINGS (2008), ao passar do tempo os ataques ficam mais poderosos e o conhecimento para executá-lo fica mais fácil de acessar.

Segundo VASCONCELOS (2015), as empresas buscam proteger seus ativos de tecnologia da informação implementando sistemas de segurança em suas redes, como: firewalls, Intrusion Detection System(IDS) e entre outros, porém a eficácia desses componentes não é suficiente, fazendo com que testes de penetração seguindo as metodologias adequadas e que melhor se aplicam sejam imprescindíveis para identificação de vulnerabilidades.

Atualmente, na área de teste de penetração há uma imensa gama de metodologias para realização de testes, é necessário verificar e experimentar qual melhor se adequa a cada caso, o presente trabalho tem por objetivo a demonstração de uma das metodologias para aplicações *web*, a metodologia OWASP (Open Web Application Security Project) que é mantida por uma organização sem fins lucrativos e tem como papel principal disseminar conhecimento para que o mundo digital se torne mais conscientizado e seguro.

A OWASP disponibiliza um guia com as dez vulnerabilidades mais comuns em servidores e aplicações *web* mapeando os ataques comuns e os pontos fracos atuais.(OWASP, 2021).

2 REVISÃO BIBLIOGRÁFICA

Desde a sua criação, a internet exerce um papel importantíssimo. Este serviço tem sido aproveitado para diversas finalidades e possui inúmeras motivações; uma delas é devido ao uso da web e a forma de distribuição de conteúdo, que deixou de ser exclusiva de cientistas e possibilitou a troca de informações entre pessoas comuns, de acordo com TAHA (2017).

Com o aumento das operações de dados na internet, avançaram também os criminosos, que utilizam desse meio para dar golpes. Para PACHECO (2018, p. 6): “A evolução social e a tecnologia proporcionaram a inovação do crime.”

Segundo JUNIOR (2021), visando esses acontecimentos, foram desenvolvidas técnicas para que as empresas conseguissem mitigar os danos causados por uma possível vulnerabilidade, realizando um teste de penetração, no qual difere do teste de software, onde é validado se o sistema segue o fluxo determinado. O teste de invasão verifica justamente se a expertise de um usuário pode ser utilizada para abusar de alguma falha.

Com todo esse conhecimento acerca do assunto, ainda é possível se deparar com

várias barreiras que dificultam o processo, na área de segurança da informação como um todo é notório a presença de várias variáveis que limitam o escopo de teste, o tempo de realização e a metodologia utilizada.

"O escopo de seus testes de invasão irá variar de cliente para cliente, assim como ocorrerá com suas tarefas."(Weidman, 2014)

Conforme elucidado por BANKS e CARRIC (2008), as empresas buscavam uma melhora na segurança da instituição investindo em testes de penetração, porém inicialmente não se seguia uma metodologia para realização desses testes, resultando em insegurança por parte das instituições tendo em vista resultados diferentes e inconsistentes.

É importante ressaltar a valia que o teste de penetração trará seguindo uma metodologia adequada ao caso, reportando sempre ao cliente através de um relatório para que sejam realizadas as correções e mitigações referentes as vulnerabilidades encontradas, explica MORENO (2015).

A metodologia utilizada se faz importante desde o desenvolvimento até o teste de segurança e a metodologia OWASP abrange todo esse processo, com o objetivo de conscientização para um aumento de softwares seguros na era digital, conforme SEDEK e OSMAN (2009).

A metodologia da OWASP é referência para pesquisadores e auditores por ser um padrão reconhecido para encontrar falhas e compreender através das análises as vulnerabilidades em ambiente web, segundo SAMPAIO (2021).

Segundo o relatório publicado pela Akamai Technologies, empresa que oferece soluções para proteção cibernética, o Brasil apresentou aumento de 180% no número de ataques a aplicações WEB entre os anos de 2019 a 2021, superando a quantia de 18 milhões alcançada em 2020.

O CERT.br(Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil) no ano de 2020 contabilizou cerca de 665.079 incidentes reportados, entre eles está uma parte relacionada a servidores e aplicações web.

O presente artigo desenvolverá um teste de intrusão utilizando a metodologia pela OWASP em um ambiente web, seguindo as etapas propostas pela metodologia, resultando em uma apresentação das etapas e das vulnerabilidades abordadas em um caso real. O objetivo é verificar se aplicações, redes e sistemas são vulneráveis ou possuem um risco de segurança não aceitável que poderia permitir o acesso não autorizado a indivíduos desconhecidos, consiste normalmente no uso de ferramentas automatizadas e técnicas manuais para testar os recursos da companhia.

3 METODOLOGIA DO TRABALHO

Foi realizado o teste de intrusão utilizando a metodologia OWASP em um ambiente web, utilizando uma abordagem em formato Caixa Preta, no qual por definição (CHIAVENATO, 2003) o interior do sistema não pode ser desvendado, ou seja, o teste é controlado porém simula um ataque real. A divisão das etapas de pesquisa foram relacionadas em três tópicos, relacionados a seguir.

3.1 IDENTIFICAÇÃO DAS VULNERABILIDADES

Nesta fase, são realizados os escaneamentos com o objetivo de coletar informações relacionadas a rede interna do cliente, sendo esta fase essencial para o prosseguimento do teste, pois, são estas informações que guiam o processo da fase seguinte.

3.2 ENUMERAÇÃO DAS VULNERABILIDADES

Nesta etapa, o objetivo é obter um mapa de exploração e risco do ambiente que será posteriormente utilizado na exploração dos ativos, que é atingido através da combinação de técnicas manuais e ferramentas automatizadas para a enumeração de informações sobre os ativos em escopo e suas vulnerabilidades.

As vulnerabilidades encontradas são classificadas pela sua severidade, de acordo com seu impacto na integridade do sistema em questão, utilizando como critério o CVSS(Common Vulnerability Scoring System), segundo SANTOS (2018) essa metodologia é gerenciada pelo FIRST e tem como objetivo ajudar as empresas a priorizar as vulnerabilidades, sendo que já foi revisada duas vezes, estando em sua terceira versão.

Para nortear este trecho do teste de penetração, foram utilizados os parâmetros de classificação do top 10 da OWASP (2021), que apresenta os dez riscos mais críticos em relação a aplicações web, segundo NETO (2019).

Tabela 1 : Top 10 Vulnerabilidades.

Categoria (OWASP)	Nome	Taxa Média de Ocorrência em aplicações web
A1	Quebra de Controle de Acesso	3.81%
A2	Falhas de Criptografia	4.49%
A3	Injeção de Código	3.37%
A4	Design não Seguro	3.00%
A5	Configuração de Segurança Incorreta	4.51%
A6	Componentes Vulneráveis e Ultrapassados	8.77%
A7	Falhas de Autenticação e Identificação	2.55%
A8	Falhas de Integridade de Dados e Software	2.05%
A9	Falhas de Monitoração e Logging	6.51%
A10	Falsificação de Requisição de Servidor ou SSRF	2.72%

Fonte: Top Ten Owasp (2021)

3.3 EXPLORAÇÃO DAS VULNERABILIDADES :

Utilizando todos os insumos coletados nas fases anteriores, o especialista busca explorar as vulnerabilidades presentes nos ativos de maneira controlada simulando um ataque lateral real com o objetivo de obter acesso aos sistemas informáticos vulneráveis e as informações ali presentes, demonstrando o que poderá ocorrer caso um atacante realize os mesmos ataques que o especialista.

É importante saber como um atacante pode se beneficiar com cada vulnerabilidade explorada. Nesse quesito, a exploração de vulnerabilidade tem como objetivo descrever fatores que compreendam os objetivos gerais dos ataques realizados na visão dos atacantes.

Nesse item será feito um resumo das principais vulnerabilidades listadas no top 10 da OWASP, como forma de estabelecer uma base para os resultados encontrados no estudo de caso. Alguns tópicos serão estruturados de forma explicativa enquanto outros serão descritos de forma mais direta, dessa forma algumas vulnerabilidades com mais aspectos a serem explorados serão contempladas igualmente, assim sendo, segue-se.

3.3.1 A1 - Quebra de controle de acesso

Falhas nesse quesito podem levar a divulgação de informações sigilosas, modificação ou destruição de dados, e ainda ao controle do acesso de funções não autorizadas por usuários comuns. São fraquezas decorrentes dessa falha:

- Violação do princípio de privilégio, onde informações particulares são divulgadas a qualquer um.
- Acesso à conta de outra pessoa, decorrente da visualização do seu identificador único.
- Elevação de privilégios, podendo executar operações de usuário sem estar ligado ou de administrador estando logado como usuário.
- Manipulação de metadados.
- Acesso através de origens desconhecidas.
- Acesso a páginas autenticadas como usuário não autenticado e a páginas restritas como usuário comum.

Mitigação :

É necessário que haja uma codificação confiável no lado do servidor em questão,

havendo também a opção de criar uma API com servidor abstrato, onde o atacante não poderá modificar o controle de acesso ou os metadados.

Algumas das maneiras de prevenir esse tipo de ataque são:

- Negar o acesso a toda informação que não seja pública.
- Implementação de mais de um mecanismo de controle de acesso para a mesma aplicação.
- Tornar o controle do poder de modificação de arquivos mais rigoroso.
- Reforçar os limites de utilização de aplicações.
- Desabilitar a listagem de diretório de servidores web e assegurar que os metadados e arquivos de backup não estejam presentes na web.
- Alertar administrador sobre falhas de controle de acesso,
- Limitar a tentativa de acesso contínuo para minimizar o ataque de ferramentas automatizadas, identificadores de sessões completas devem ser invalidados após logout.

3.3.2 A2 - Falhas de Criptografia

Falhas de criptografia podem levar a exposição de dados sensíveis. É necessário que se determine que dados, tanto estáticos quanto em trânsito, precisam de tratamento especial.

Algumas análises que servem para identificar essas fraquezas são:

- Há algum dado sendo transmitido em texto claro.
- Os algoritmos e protocolos utilizados são ultrapassados ou fracos.
- Há alguma chave criptográfica padrão sendo utilizada, ou se é fraca ou reutilizada.
- A encriptação foi mesmo implantada.
- Se há algum *header* HTTP faltando no navegador.
- Os certificados recebidos foram propriamente avaliados.
- Algum vetor de inicialização foi ignorado, reutilizado ou não propriamente gerado para o modo criptográfico implantado.
- Senhas estão sendo utilizadas como chaves criptográficas na ausência de um algoritmo gerador de senhas.
- O algoritmo de randomização utilizado para encriptação é forte o suficiente.
- Os métodos e funções utilizados são adequados à função de criptografia.

Mitigação :

É importante classificar todos os dados necessários de acordo com sua importância, apagando aqueles que são sensíveis e que não precisam estar armazenados e encriptando todos os que precisam.

Desabilitar o cache para interações que acessam dados sensíveis, ao mesmo passo que se utiliza protocolos adequados para o seu transporte, em detrimento de protocolos ultrapassados.

3.3.3 A3 - Injeção de Código

A injeção trata-se da inserção de dados para facilitação do uso de comandos mal-intencionados. Esses ataques têm uma dependência das tecnologias vigentes.

Algumas falhas que tornam a aplicação suscetível à injeção são :

- Falta de validação e filtro para as informações fornecidas pelo usuário.
- Chamadas sem validação de contexto.
- Dados maliciosos sendo utilizados para extrair outros dados sensíveis do sistema.
- Injeção de SQL, que é utilizada para extrair informações de consultas SQL.

Mitigação:

A principal forma de prevenir injeções de código é manter dados de comandos e consultas separados, utilizando também controles SQL como forma de

prevenir um vazamento de dados caso haja injeção de SQL.

3.3.4 A4 - Design não seguro

O design não seguro trata de diferentes tipos de fraquezas, relacionadas à projeção, que tornam um aplicativo inseguro, não necessariamente fazendo menção às outras vulnerabilidades apresentadas pela OWASP. O design não seguro não decorre de uma implementação ruim, e sim de uma falta de conhecimento sobre que tipo de seguranças são necessárias para determinada aplicação.

Mitigação :

É necessário que se faça uma análise de todos os parâmetros e dados presentes na aplicação, os quais deverão ser trabalhados de forma condizente com sua importância e confidencialidade.

Um design seguro é alcançado através de uma metodologia que avalia constantemente a proteção contra ataques já conhecidos, sempre refinando a proteção contra os riscos mais atuais e sempre buscando a correção de erros encontrados desde o início.

3.3.5 A5 - Configuração de segurança incorreta

Aqui serão demonstrados alguns dos erros nas configurações de segurança dos aplicativos web que os tornam vulneráveis.

- Falta de robustez de segurança em qualquer parte da aplicação ou configurações de permissão mal-implementadas em serviços de nuvem.
- Ferramentas desnecessárias habilitadas.
- Contas padrão ainda habilitadas e não modificadas.
- Mensagens de erro que detalham muito sobre o sistema para os usuários, e que podem acabar revelando informações que facilitem a intrusão.
- Últimas atualizações de segurança do sistema desabilitadas ou não configuradas corretamente.

Mitigação :

Tornar a aplicação mais limpa, diminuindo a quantidade de componentes em função da qualidade e buscar verificar sempre a efetividade das configurações, seja de forma manual ou automática. Outra prevenção importante é sempre manter os usuários informados sobre questões de segurança, como mudança de senhas e utilização de senhas diferentes para diferentes aplicações.

3.3.6 A6 - Componentes vulneráveis e ultrapassados

Uma das principais causas são aplicações com componentes desatualizados. Isso acontece geralmente quando não se sabe as versões de todos os componentes usados, assim como suas dependências e, por isso, se deixa de corrigir o erro. Outra causa é a demora na atualização dos componentes, principalmente provocada por sistemas onde é determinado um tempo específico, que pode ser longo demais, para que se procure ou instale uma atualização de segurança.

Mitigação :

Não utilizar componentes desnecessários. Manter um plano de ação que seja capaz de constantemente detectar as vulnerabilidades atuais e saná-las o mais rápido possível.

3.3.7 A7 - Falhas de autenticação e identificação :

As falhas nesse âmbito ocorrem quando o atacante consegue de alguma forma, entrar em uma sessão como se fosse um usuário habitual. As principais causas estão em sistemas que permitem ataques de força bruta, ataques de violação de dados, tem processos de recuperação de senhas fracos, e

possuem fraquezas relacionadas à sua má construção.

Mitigação :

Principalmente a utilização de autenticação multifator, além da limitação de quantidade de tentativas de login erradas. Utilizar também fatores que tornam mais difícil a criação de uma senha fraca, aliadas com uma camuflagem por parte do servidor, que pode gerar um número de identificação randômico para identificação depois do login do usuário.

3.3.8 A8 - Falhas de integridade de dados e software :

Aqui são tratadas principalmente falhas de código e infraestrutura. Algumas das principais causas estão relacionadas a aplicações que dependem de módulos de terceiros, muitos dos quais derivam de fontes inseguras.

Mitigação :

Verificação de assinatura digital para saber se os dados de fontes diversas são seguros e não foram alterados.

Evitar que dados não encriptados escapem para clientes não verificados.

Revisão de código e configurações para redução de risco de ataques.

3.3.9 A9 - Falhas de monitoração e registro :

Acontece quando eventos importantes ocorridos no sistema não são registrados, ou são parcialmente registrados ou até registrados incorretamente. Alguns exemplos de falhas desse tipo são :

- Os registros não estão armazenados localmente.
- Alertas não são gerados adequadamente, e os processos de resposta aos erros são inefetivos.

Mitigação :

Todos os registros devem ser adequadamente armazenados e suas mensagens devem deixar claro aos solucionadores de erros os problemas encontrados, para que a soluções escolhidas sejam efetivadas.

3.3.10 A10 - Falsificação de requisição de servidor ou SSRF :

Esse tipo de ataque ocorre quando a aplicação não valida a URL, permitindo que os recursos sejam compartilhados sem maiores problemas. Dessa forma o atacante poderá forçar a aplicação a mandar solicitações em fontes desconhecidas, mesmo com proteções ativas.

Mitigação :

Na camada de rede é importante que se divida o acesso remoto em diferentes redes, além de reforçar as políticas de segurança dos firewalls e as regras de acesso à rede, principalmente em redes locais onde somente o essencial precisa ser acessado. Manter os registros de erros nos firewalls também é de extrema importância.

Já na camada de aplicação é relevante tratar todos os dados inseridos, validando as entradas dos clientes, desabilitando redirecionamentos HTTP quando não forem necessários e evitando mandar respostas desnecessárias aos clientes. Também se deve ter cuidado ao usar listas de expressões já conhecidas para mitigar a SSRF, uma vez que os atacantes têm meios para burlar esse sistema.

4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

4.1 PREPARAÇÃO

4.1.1 Definição de escopo

Durante a definição do escopo, foi convencionado que o teste deveria ser realizado em ambiente de testes, em uma aplicação que utiliza o software GPWeb, que é responsável pela gestão estratégica e gerenciamento de projetos orientados a processos de padrão mundial. O teste de

penetração foi realizado na modalidade caixa preta, significando que a equipe que realizará o teste teve acesso restrito às informações do sistema, simulando um atacante real.

As ferramentas utilizadas na identificação das vulnerabilidades foram as seguintes.

- Wappalyzer :

Uma extensão para os navegadores Firefox e Chrome usada para identificar as tecnologias utilizadas em um site, como por exemplo a plataforma do site, o sistema operacional do servidor, dentre outras.

- BurpSuite :

Ferramenta usada para realizar testes de segurança em aplicações web, utilizando um proxy para registrar requisições e retornos dos dados que estão tramitando entre o navegador e a aplicação web.

- SQLMap :

Ferramenta de código aberto que torna o processo de detectar e explorar falhas de injeção SQL automático, permitindo assumir o controle dos bancos de dados de aplicações web vulneráveis.

4.2 RECONHECIMENTO

Utilizando o plugin Wappalyzer (versão 5.8.2), plugin utilizado para identificar as tecnologias utilizadas em um site, no navegador Firefox foram identificadas as seguintes informações relativas ao sistema já disponíveis em fontes abertas:

- Linguagem de programação PHP;
- Sistema Operacional Debian;
- Servidor Web Apache 2.4.10; e
- Biblioteca JavaScript jQuery 1.9.1

Além disso, foi identificado que existem vulnerabilidades conhecidas e catalogadas para o sistema GPWeb. Foram identificados os seguintes CVE (Common Vulnerabilities and Exposures), na versão 8.4.61 do GPWeb:

- **CVE-2017-15877** – Permissões inseguras no arquivo db.php;
- **CVE-2017-15875** - SQL injection no parâmetro *"checkemail"* da função de recuperar senha;
- **CVE-2017-15876** – Permissão irrestrita de upload de arquivos, permitindo o envio de qualquer tipo de arquivo.

4.3 ESCANEAMENTO DE VULNERABILIDADES

Foram realizadas varreduras manuais e automatizadas, com as ferramentas para pentest, BurpSuite e SQLMap, visando buscar as vulnerabilidades presentes no sistema. As falhas encontradas estão relatadas no próximo item.

4.4 ANÁLISE DE VULNERABILIDADES

A severidade apresentada para as vulnerabilidades segue o modelo proposto no Common Vulnerability Scoring System (CVSS) versão 3.0. A identificação das vulnerabilidades é apresentada conforme o Common Weakness Enumeration (CWE).

Quadro 1: Vulnerabilidades encontradas.

Categoria (OWASP)	Severidade	Vulnerabilidade
A5	Crítica	Upload irrestrito de arquivos com tipo perigoso (CWE-434)
A3	Crítica	Injeção de SQL (CWE-89)
A1	Crítica	Quebra de controle de autenticação (CWE-640)
A7	Alta	Ataque de força bruta (CWE-307)
A4	Baixa	Clickjacking (CWE-1021)

Fonte: Top Ten Owasp (2021).

4.4.1 Injeção de SQL (CWE-89)

4.4.1.1 Descrição

A injeção de SQL (SQL Injection) consiste numa técnica de ataque onde o atacante manipula o código SQL, que é a linguagem utilizada para troca de informações entre aplicativos e bancos de

dados relacionais, onde é possível obter dados através de uma falha na codificação da aplicação.
Severidade: Crítica

4.4.1.2 Impacto

Um invasor pode executar instruções SQL arbitrárias no sistema. Isso pode comprometer a integridade do banco de dados e/ou expor informações a usuários que não deveriam ter acesso a elas. Dependendo do banco de dados em uso, as vulnerabilidades de injeção de SQL levam a diferentes níveis de acesso de sistema para o invasor. Pode ser possível ler ou escrever em arquivos, ou executar comandos de terminal no sistema operacional subjacente. Inclusive, se um invasor obtiver acesso a certos procedimentos do banco de dados, poderá comprometer toda a máquina.

4.4.1.3 Exploração

Utilizando a ferramenta *sqlmap* foi possível explorar a vulnerabilidade da aplicação através do parâmetro *checkemail* no formulário de login da página. A exploração da vulnerabilidade permitiu obter acesso ao nome do banco de dados utilizado pela aplicação. A Figura 1 é parte das informações extraídas do banco de dados durante a exploração.

Figura 1: Nome do banco de dados.

```
[08:59:28] [ERROR] unable to retrieve the number of databases
[08:59:28] [INFO] falling back to current database
[08:59:28] [INFO] fetching current database
[08:59:29] [INFO] retrieved: 'gp    _prod'
available databases [1]:
[*] gp    _prod
```

Fonte: Autor.

4.4.1.4 Mitigações sugeridas

Os parâmetros enviados para o banco de dados devem sofrer processo de sanitização contra SQL Injection, como por exemplo, filtrar metacaracteres da entrada do usuário.

4.4.2 Ataque de força bruta (CWE-307)

4.4.2.1 Descrição

Um ataque de força bruta é utilizado quando um invasor busca desvendar uma senha ou uma chave de criptografia. Esse tipo de ataque é realizado da seguinte forma : um invasor tenta, utilizando algum método sistemático disponível, descobrir a informação fazendo testes de combinação com todas as possibilidades de caracteres disponíveis para a referida senha ou chave, utilizando também uma alternativa que se denomina ataque de dicionário, que se trata de uma técnica que utiliza uma lista de senhas populares ou que foram expostas de alguma forma como teste para realizar o ataque. Um ataque de força bruta bem sucedido permite que um invasor obtenha credenciais de usuários válidas.
Severidade: Alta

4.4.2.2 Impacto

O sistema permite que um usuário mal-intencionado realize diversas requisições com diferentes valores para encontrar um token válido e recuperar documentos que estão disponibilizados na aplicação.

4.4.2.3 Exploração

O sistema permitiu realizar várias tentativas de login, com utilização de uma ferramenta automatizada para força bruta do *Burp Suite*, conforme a Figura 2. A efetividade do ataque dependerá do dicionário utilizado e no caso deste sistema, a política de senhas permite a criação de senhas fracas, dedutíveis em um ataque.

Figura 2: Ataque de força bruta.

Request	Payload1	Payload2	Status	Error	Timeout	Length	Comment
7	teste	teste	200			5776	
8	jose	teste	200			5776	
9	hacker	teste	200			5776	
10	std	teste	200			5776	
11	admin	jose	200			5776	
12	teste	jose	200			5776	
13	jose	jose	200			5776	
14	hacker	jose	200			5776	
15	std	jose	200			5776	
16	admin	123456	200			80737	
18	jose	123456	200			9602	
19	hacker	123456	200			5776	
20	std	123456	200			5776	

Fonte: Autor.

Como pode ser observado, as requisições 16 e 18, destacadas na Figura 2, correspondem a senhas válidas para os usuários especificados. O sucesso do ataque pode ser percebido através do length diferente de 5776 na resposta do servidor.

4.4.2.4 Mitigações sugeridas

Implementar captchas ou medida equivalente para se impedir ataques de força bruta.

4.4.3 Quebra de controle de autenticação (CWE-640)

4.4.3.1 Descrição

Falhas nesse quesito podem levar a divulgação de informações sigilosas, modificação ou destruição de dados, e ainda ao controle do acesso de funções não autorizadas por usuários comuns.
Severidade: Crítica

4.4.3.2 Impacto

O atacante pode ser um agente externo anônimo e/ou utilizadores internos. É permitido alterar a senha de outros usuários. Sendo o ataque bem-sucedido, o atacante poderá fazer tudo como se fosse a vítima, podendo assim ter uma conta de acesso com maiores privilégios.

4.4.3.3 Exploração

É possível realizar a troca de senha de qualquer usuário do sistema a partir do mecanismo de troca de senhas do GPweb. Para isso, basta enviar a requisição da Figura 3.

Figura 3: Requisição para troca de senha.

```
1 POST /[redacted]/server/index.php HTTP/1.1
2 Host: 172.[redacted]
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://172.[redacted]server/index.php
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 103
0 Connection: close
1 Upgrade-Insecure-Requests: 1
2
3 celular=0&perdeu_senha=1&pesquisa_email=1&pesquisa_frase=0&gravar_senha=0&usuario_id=0&checkemail=gpweb
```

Fonte: Autor.

Após essa requisição, será exibido na tela um campo onde pode-se informar qualquer usuário do sistema (Figura 4).

Figura 4: Formulário para nome.

Fonte: Autor.

Ao clicar em buscar será aberta uma caixa de interrogação. Caso a pergunta de segurança cadastrada no banco de dados esteja vazia, basta também informar a resposta vazia e será concedido acesso à caixa para troca de senha. O formulário contém uma validação do lado do cliente do conteúdo. Para contornar essa validação, desabilite o javascript do browser ou envie qualquer conteúdo e faça a alteração no burp suite. A Figura 5 exibe o formulário para responder a pergunta de segurança.

Figura 5: Formulário para resposta da pergunta de segurança.

Fonte: Autor.

Com isso feito, a Figura 6 exibe o formulário para a troca de senha do usuário.

Figura 6: Troca de senha de usuário.

Fonte: Autor.

Após esse procedimento, foi trocada a senha do usuário admin para 123456. Agora basta realizar o login no sistema.

4.4.3.4 Mitigações sugeridas

Não permitir o cadastro de perguntas e respostas de segurança vazias. E realizar a validação de dados do lado do servidor.

Vale ressaltar que essa vulnerabilidade se vale de funcionalidades legadas do GPWeb que, a princípio, não deveriam constar no código atual da aplicação.

4.4.4 Clickjacking: ausência de cabeçalho do X-Frame-Options (CWE-1021)

4.4.4.1 Descrição

Clickjacking é uma técnica maliciosa de enganar um usuário para clicar em algo diferente do que ele percebe que está clicando, potencialmente revelando informações confidenciais ou controlando o computador clicando em páginas da Web aparentemente inócuas.
Severidade: Baixa.

4.4.4.2 Impacto

O clickjacking ocorre quando um invasor utiliza camadas “invisíveis” para induzir o usuário a clicar em um botão ou link de uma página em seu domínio. Dessa forma o invasor faz com que os cliques destinados a uma página sejam encaminhados para outra.

É possível também que o invasor utilize uma técnica semelhante para fazer com que um usuário acredite estar digitando uma senha em seus e-mails ou contas bancárias, quando na verdade está digitando em um quadro invisível controlado pelo invasor. Esse tipo de ataque é realizado utilizando uma combinação cuidadosamente elaborada de folhas de estilo, iframes e caixas de texto.

4.4.4.3 Exploração

O servidor não retornou um cabeçalho X-Frame-Options, o que significa que este site pode estar sob o risco de um ataque de clickjacking. O cabeçalho de resposta HTTP X-FrameOptions pode ser usado para indicar se um navegador deve ou não ter permissão para renderizar uma página dentro de um frame ou iframe. Os sites podem usar isso para evitar ataques de clickjacking, garantindo que o conteúdo deles não seja incorporado a outros sites.

4.4.4.4 Mitigações sugeridas

Configurar o servidor Web para incluir um cabeçalho X-Frame-Options.

4.4.5 Upload irrestrito de arquivos com tipo perigoso (CWE-434)

4.4.5.1 Descrição

A aplicação permite que o atacante realize o upload ou transferência de arquivos com tipo perigoso, que podem ser automaticamente processados no ambiente de produção.
Severidade: Crítica.

4.4.5.2 Impacto

A execução arbitrária de código é possível caso um arquivo enviado seja interpretado e seu código executado pelo servidor. Isso é particularmente válido para extensões .asp e .php enviadas para servidores web devido a estes arquivos serem comumente tratados com execução automática, mesmo quando as permissões do sistema de arquivos não determinam a execução. Por exemplo, em ambientes Unix, programas não podem, normalmente, ser executados a menos que o bit de execução esteja setado. Porém, programas PHP podem ser executados pelo servidor web sem que sejam diretamente invocados pelo sistema operacional.

4.4.5.3 Exploração

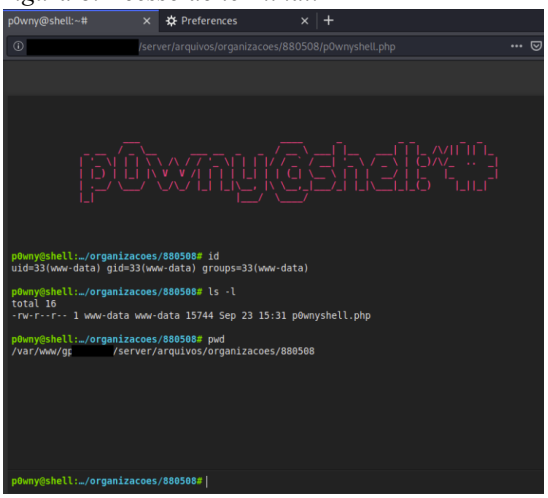
É possível realizar o upload de um arquivo .php utilizando a funcionalidade que define um logotipo para a organização, exibido na Figura 7.

Figura 7: Upload do arquivo no logotipo.

Fonte: Autor.

O arquivo é salvo no diretório "/server/arquivos/organizacoes/id/" e acessível diretamente pelo browser. O acesso ao arquivo php enviado dá acesso a um web terminal na máquina que hospeda o sistema, sendo possível visualizar todos arquivos sensíveis que ali estão, como na Figura 8.

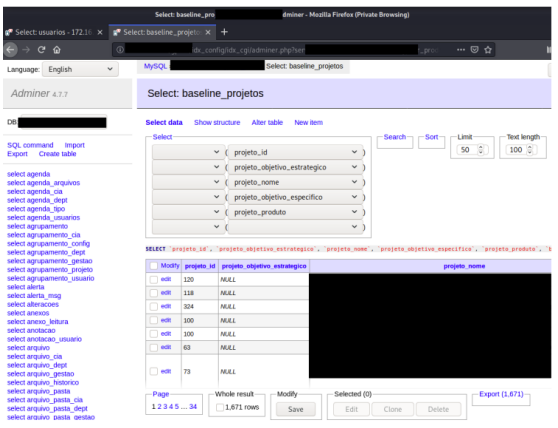
Figura 8: Acesso ao terminal.



Fonte: Autor.

Como o sistema permite o envio de arquivos PHP e, tendo acesso aos dados de conexão com o banco de dados, foi possível enviar a aplicação "Adminer" (https://www.adminer.org/), escrita em PHP, fornecendo uma interface de acesso e gerenciamento ao banco de dados, conforme a Figura 9.

Figura 9: Acesso ao gerenciador de banco de dados.



Fonte: Autor.

4.4.5.4 Mitigações sugeridas

Configurar a aplicação para receber apenas as extensões permitidas para logo.

5 CONSIDERAÇÕES FINAIS

Foram encontradas diversas vulnerabilidades na aplicação, tornando elevado o risco de hospedagem e utilização do sistema. A exploração de algumas das

vulnerabilidades apontadas poderia permitir o acesso do atacante ao sistema operacional da máquina que o hospeda, colocando em risco toda a infraestrutura do ambiente. Além disso, a exploração das vulnerabilidades aqui detalhadas pode permitir ao atacante o acesso indiscriminado a todos os dados armazenados no banco de dados do sistema. Vale ressaltar também que, diversas falhas identificadas têm sua origem no código original do sistema GPWeb. Recomendamos que as funcionalidades legadas, que não sejam utilizadas no sistema, sejam excluídas do código da aplicação atual. Além disso, como o sistema atual é fortemente baseado em versões antigas do GPWeb, é altamente recomendado que se acompanhe as atualizações de segurança aplicadas ao sistema original para que se analise a pertinência das mesmas correções no sistema. Por fim, ressaltamos que a última versão do GPWeb (disponível em : https://softwarepublico.gov.br/social/gpweb), é a de número 8.5.22.

Além das mitigações sugeridas, há boas práticas que podem ser adotadas para o sistema em questão. É importante que as flags HTTPOnly e SecureFlag sejam habilitadas no sistema. Quando o HTTPOnly está ativo, o sequestro de sessão através de XSS torna-se mais difícil de implementar. Ao ativar o SecureFlag, por sua vez, o navegador impedirá a transmissão dos cookies por um canal que não contenha criptografia.

Por fim, recomendamos que, a cada alteração significativa nos componentes do sistema, seja realizado um novo teste de segurança a fim de manter a sua confidencialidade, integridade e disponibilidade.

Quaisquer informações relacionadas aos testes de análise de vulnerabilidades do sistema deverão tramitar em meio seguro, incluindo informações contidas neste relatório e informações do sistema, por se tratar de conteúdo de acesso restrito.

Apesar de ser perceptível que as incidências de vulnerabilidades se manifestam em parte, por falhas relacionadas à programação das aplicações, segundo o olhar estatístico sobre os principais tipos de ataques ocorridos em aplicações web, nota-se também que a construção dessas aplicações como um todo, desde seu design, passando pelos componentes utilizados na sua construção, e até a maneira como os usuários se comportam ao usá-las, importa para que o resultado final seja um ambiente o mais livre possível de ameaças.

REFERÊNCIAS

[1] BANKS, Taylor; CARRIC. **The Pentest is dead, long live the Pentest!** DEF CON® 16 Hacking Conference, 2008. 77 f. Disponível em: <https://defcon.org/images/defcon-16/dc16-presentations/defcon-16-banks-carric.pdf>.

[2] CHIAVENATO, Idalberto. **Teoria geral da administração**. 6. ed. Rio de Janeiro : Editora Elsevier, 2002.

[3] CVE-2017-15877, **Common Vulnerabilities and Exposures**, 2022, Disponível em:<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-15877>. Acesso em : 5 de jun. de 2022.

[4] CVE-2017-15875, **Common Vulnerabilities and Exposures**, 2022, Disponível em : <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-15875>. Acesso em : 5 de jun. de 2022.

[5] CVE-2017-15876, **Common Vulnerabilities and Exposures**, 2022, Disponível em : <https://cve.mitre.org/cgi-bin/cvename.cgi?name=2017-15876>. Acesso em : 5 de jun. de 2022.

[6] COMMON VULNERABILITY SCORING SYSTEM. **First**, 2022. Common Vulnerability Scoring System v3.1: Specification Document. Disponível em : <https://www.first.org/cvss/>. Acesso em : 5 de jun. de 2022.

[7] CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection'), **Common Weakness Enumeration**, 2022, Disponível em : <https://cwe.mitre.org/data/definitions/89.html>. Acesso em : 5 de jun. de 2022.

[8] CWE-307: Improper Restriction of Excessive Authentication Attempts, **Common Weakness Enumeration**, 2022, Disponível em : <https://cwe.mitre.org/data/definitions/307.html>. Acesso em : 5 de jun. de 2022.

[9] CWE-640: Weak Password Recovery Mechanism for Forgotten Password, **Common Weakness Enumeration**, 2022, Disponível em : <https://cwe.mitre.org/data/definitions/640.html>. Acesso em : 5 de jun. de 2022.

[10] CWE-1021: Improper Restriction of Rendered UI Layers or Frames, **Common Weakness Enumeration**, 2022, Disponível em : <https://cwe.mitre.org/data/definitions/1021.html>. Acesso em : 5 de jun. de 2022.

[11] JÚNIOR, José. Livro de Pentest. **Casa do código**, 2021. Disponível em : <https://www.casadocodigo.com.br/products/livro-pentest>. Acesso em : 25 mar. de 2022.

[12] MORENO, Daniel. **Introdução ao Pentest**. 1. ed. São Paulo : Editora Novatec, 2019.

[13] NETO, David. **Web (eternamente) revisitada :** análise de vulnerabilidades web e de ferramentas de código aberto para exploração. 2019. 92 f. Dissertação (Ciência da Computação) - Graduação em Informática - Universidade Federal do Paraná, Curitiba, 2019.

[14] OWASP TOP 10 TEAM. **Owasp**, 2021. Welcome to OWASP top 10. Disponível em : <https://owasp.org/Top10/>. Acesso em : 5 de jun. de 2022.

[15] PACHECO, Gisele; COSTA, Renato. **Crimes virtuais e legislação penal brasileira**. Revista eletrônica de ciências jurídicas, Ipatinga, v. 1. n. 1. p. (1-39), janeiro, 2018.

[16] SAMPAIO, Felipe. **Uma análise prática das principais vulnerabilidades em aplicações WEB baseado no top 10 OWASP**. 2021. 61 f. Graduação em Redes de Computadores - Universidade Federal do Ceará, Quixada, 2021.

[17] SANTOS, Vanessa. **Melhoria ao sistema de avaliação de vulnerabilidades - CVSS**. 2018. 139 f. Dissertação (Mestrado em Segurança Informática) - Universidade de Lisboa, Lisboa, 2018.

[18] SEDEK, Khairul; OSMAN, Norlis; OSMAN, Mohd; JUSOFF, Hj. **Developing a Secure Web Application Using OWASP Guidelines**. CCSE, Perlis, v. 2. n. 4. p. (1-7), novembro, 2009.

[19] STALLINGS, William. **Criptografia e segurança de redes: Princípios e práticas**. 4. ed. São Paulo: Pearson Prentice Hall, 2008. 481 p.

[20] TAHA, Antonio. **Guia de testes de segurança para aplicações web**. 2017. 127 f. Monografia (Curso de Sistemas de Informação) - Universidade Federal de Santa Catarina, Florianópolis, 2017.

[21] WEIDMAN, Georgia. **Testes de Invasão:** Uma introdução prática ao hacking. 1. ed. São Paulo: Editora Novatec, 2014.