



MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
DE GOIÁS - CÂMPUS URUAÇU

GILTON CUNHA DE JESUS

SEGURANÇA DA INFORMAÇÃO EM APLICAÇÕES WEB

URUAÇU
2022



INSTITUTO FEDERAL
Goiás

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
SISTEMA INTEGRADO DE BIBLIOTECAS

TERMO DE AUTORIZAÇÃO PARA DISPONIBILIZAÇÃO NO REPOSITÓRIO DIGITAL DO IFG - ReDi IFG

Com base no disposto na Lei Federal nº 9.610/98, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia de Goiás, a disponibilizar gratuitamente o documento no Repositório Digital (ReDi IFG), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, em formato digital para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IFG.

Identificação da Produção Técnico-Científica

- | | |
|--|---|
| ☐ Tese | ☐ Artigo Científico |
| ☐ Dissertação | ☐ Capítulo de Livro |
| ☐ Monografia – Especialização | ☐ Livro |
| ☒ TCC - Graduação | ☐ Trabalho Apresentado em Evento |
| ☐ Produto Técnico e Educacional - Tipo: _____ | |

Nome Completo do Autor: Gilton Cunha de Jesus

Matrícula: 20161050100138

Título do Trabalho: SEGURANÇA DA INFORMAÇÃO EM APLICAÇÕES WEB

Autorização - Marque uma das opções

1. (X) Autorizo disponibilizar meu trabalho no Repositório Digital do IFG (acesso aberto);
2. () Autorizo disponibilizar meu trabalho no Repositório Digital do IFG somente após a data ____/____/____ (Embargo);
3. () Não autorizo disponibilizar meu trabalho no Repositório Digital do IFG (acesso restrito).

Ao indicar a opção **2 ou 3**, marque a justificativa:

- () O documento está sujeito a registro de patente.
() O documento pode vir a ser publicado como livro, capítulo de livro ou artigo.
() Outra justificativa: _____

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O/A referido/a autor/a declara que:

- i. o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;
- ii. obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autor/a, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia de Goiás os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;
- iii. cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia de Goiás.

União 02/12/2022
Local Data

Gilton Cunha de Jesus
Assinatura do Autor e/ou Detentor dos Direitos Autorais



GILTON CUNHA DE JESUS

SEGURANÇA DA INFORMAÇÃO EM APLICAÇÕES WEB

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia de Goiás - Câmpus Uruaçu ligado ao Ministério da Educação, como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Área de concentração: Análise e Desenvolvimento de Sistemas

Linha de pesquisa: Segurança da Informação

Orientador: Maurílio Humberto Rodrigues Miranda
Instituto Federal de Educação, Ciência e
Tecnologia de Goiás

URUAÇU
2022

Dados Internacionais de Catalogação na Publicação (CIP)

Jesus, Gilton Cunha de.

J58s Segurança da informação em aplicações web [manuscrito] / Gilton Cunha de
Jesus. - 2022.
LX, 60 f.: il.

Orientador: Prof. Me. Maurílio Humberto Rodrigues Miranda.

Trabalho de Conclusão de Curso (Graduação) - Instituto Federal de Educação,
Ciência e Tecnologia de Goiás: Campus Uruaçu, Curso de Tecnologia em Análise e
Desenvolvimento de Sistemas, 2022.

Bibliografia.

Inclui lista de figuras, lista de siglas e figuras.

1. Segurança da informação. 2. Aplicação web. 3. OWASP. I. Miranda, Maurílio
Humberto Rodrigues (orientador). II. Instituto Federal de Educação, Ciência e
Tecnologia de Goiás. III. Título.

CDD: 005.8

Ficha catalográfica elaborada pela Bibliotecária Sabrina Gisele da Silva Felix CRB1/2561



INSTITUTO FEDERAL
Goiás

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE GOIÁS
CÂMPUS URUAÇU

Gilton Cunha de Jesus

SEGURANÇA DA INFORMAÇÃO EM APLICAÇÕES WEB

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas ao Departamento de Áreas Acadêmicas do Instituto Federal de Educação, Ciência e Tecnologia de Goiás - Campus Uruaçu.

Aprovado em 12 de novembro de 2022.

(Assinado eletronicamente)

Prof. Me. Maurílio Humberto Rodrigues Miranda
Instituto Federal de Goiás - Campus Uruaçu - Orientador

(Assinado eletronicamente)

Prof. Me. Alessandro Siqueira da Silva
Instituto Federal de Goiás - Campus Uruaçu

(Assinado eletronicamente)

Profa. Esp. Mateus Nunes dos Santos
Instituto Federal de Goiás - Campus Uruaçu

Documento assinado eletronicamente por:

- Alessandro Siqueira da Silva, PROFESSOR ENS BASICO TECN TECNOLOGICO, em 28/11/2022 19:53:36.
- Mateus Nunes dos Santos, TEC DE TECNOLOGIA DA INFORMACAO, em 28/11/2022 16:26:18.
- Maurílio Humberto Rodrigues Miranda, CHEFE - CD4 - URU-DAA, em 28/11/2022 15:58:35.

Este documento foi emitido pelo SUAP em 26/11/2022. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse <https://suap.ifg.edu.br/autenticar-documento/> e forneça os dados abaixo:

Código Verificador: 349613
Código de Autenticação: 8293467abb



AGRADECIMENTOS

A Deus, pela minha vida, e por me ajudar a ultrapassar todos os obstáculos encontrados ao longo do curso. A minha família, que me incentivaram nos momentos difíceis e compreenderam a minha ausência enquanto me dedicava à realização deste trabalho. Ao meu orientador que se absteve dos seus momentos de descanso independente do horário para me auxiliar na elaboração deste trabalho.

RESUMO

CUNHA DE JESUS, Gilton. SEGURANÇA DA INFORMAÇÃO EM APLICAÇÕES WEB. 2022. 48 f. Trabalho de Conclusão de Curso – Tecnologia em Análise e Desenvolvimento de Sistemas, Instituto Federal de Educação, Ciência e Tecnologia de Goiás - Câmpus Uruaçu. Uruaçu, 2022.

Com o avanço da tecnologia, as aplicações *web* têm se tornado cada vez mais comum no nosso meio, assim como usuários e atividades utilizando aplicações *web*. Esse aumento traz inúmeras vantagens mas também chama a atenção para potenciais ataques. A segurança de aplicações *web* é aprimorada a cada dia para mitigar as vulnerabilidades e proteger os usuários finais contra intrusão e perda de dados. Portanto este trabalho de conclusão de curso tem a proposta de mostrar as principais vulnerabilidades enumeradas pelo OWASP (*Open Web Application Security Project*), como um ataque é efetuado, seus impactos e formas de mitigação para prevenção de ataques futuros é ajudará desenvolvedores a mitigar e prevenir futuros ataques a uma aplicação *web*.

Palavras-chave: : segurança, vulnerabilidades, aplicações web, OWASP.

ABSTRACT

CUNHA DE JESUS, Gilton. Title in English. 2022. 48 f. Trabalho de Conclusão de Curso – Tecnologia em Análise e Desenvolvimento de Sistemas, Instituto Federal de Educação, Ciência e Tecnologia de Goiás - Câmpus Uruaçu. Uruaçu, 2022.

With the advancement of technology, web applications have become increasingly common. in our environment, as well as users and activities using web applications. with this increase brings numerous advantages but also draws attention to potential attacks. web application security is improved every day to mitigate vulnerabilities and protect end users from intrusion and data loss. So this work of course conclusion has the proposal to show the main vulnerabilities listed by OWASP (Open Web Application Security Project), how an attack is carried out, its impacts and ways of mitigating to future attacks. All the contents of this work help developers to mitigate and prevent future attacks on a web application.

Keywords: security, vulnerabilities, web applications, OWASP.

LISTA DE FIGURAS

Figura 1 – Vulnerabilidades CVE em anos	5
Figura 2 – Principais tipos de vulnerabilidades	6
Figura 3 – Fluxograma utilizado pelo atacante para atingir uma aplicação	10
Figura 4 – Comparação do Owasp de 2017 para 2021	11
Figura 5 – Modelo de ataque SSRF	17
Figura 6 – Classificação de risco	25
Figura 7 – Modelo de teste OWASP ZAP	26
Figura 8 – <i>Download do OWASPZap</i>	27
Figura 9 – Tela inicial do <i>OWASPZap</i>	27
Figura 10 – <i>Scanner</i> automático <i>OWASPZap</i>	28
Figura 11 – Resultado do teste	29
Figura 12 – Mozilla Firefox	30
Figura 13 – Tela de configuração de conexão Mozilla Firefox	31
Figura 14 – Configuração de proxy Mozilla Firefox	31
Figura 15 – Laboratório Acunetix	32
Figura 16 – Pacote interceptado pelo <i>Burp Suite</i>	33
Figura 17 – Início do ataque pelo Burp Suite	33
Figura 18 – Localização dos campos que serão injetados <i>payloads</i>	34
Figura 19 – Escolha do primeiro <i>payload</i>	35
Figura 20 – <i>Wordlist</i> de usuários	35
Figura 21 – Escolha do segundo <i>payload</i>	36
Figura 22 – Resultado do ataque	37
Figura 23 – Fixados campos de usuário e senha	38
Figura 24 – Acesso ao laboratório <i>Acunetix</i> realizado com sucesso	38
Figura 25 – Acesso ao laboratório Portswigger	39
Figura 26 – Inseridos campos de usuário e senha no Laboratório Portswigger	40
Figura 27 – Tela de inteceptação do Burp Suite	40
Figura 28 – <i>Burp Suite</i>	41
Figura 29 – Inserido código de injeção	42
Figura 30 – Laboratório Portswigger acessado com sucesso	42
Figura 31 – Site seguro interceptado	43

LISTA DE ABREVIATURAS E SIGLAS

OWASP	Open Web Application Security Project
APIs	Application Pro-gramming Interface
CMS	Content Management System
SQL	Structured Query Language
SO	Sistema Operacional
LDAP	Lightweight Directory Access Protocol
XPath	XML Path Language
XML	Extensible Markup Language
SMTP	Simple Mail Transfer Protocol
ORM	Object-relational mapping
URL	Uniform Resource Locator
JSON	JavaScript Object Notation
ARP	Address Resolution Protocol
IP	Internet Protocol
URI	Uniform Resource Identifier
DoS	Denial of Service
HTTPS	Hyper Text Transfer Protocol Secure
HTTP	Hyper-text Transfer Protocol
XSS	Cross-Site Scripting
RPC	Remote Procedure Call
IPC	Inter-Process Communication
HTML	HyperText Markup Language
MitM	Man in the Middle
XXE	Entidades Externas de XML

CVE	Common Vulnerabilities and Exposures
CWE	Common Weakness Enumeration

SUMÁRIO

1 – INTRODUÇÃO	1
1.1 Objetivos	2
1.1.1 Objetivos Especificos	2
1.2 Organização do Trabalho	3
2 – FUNDAMENTAÇÃO TEÓRICA	4
2.1 Segurança da Informação	4
2.2 Aplicações Web e suas Vulnerabilidades	5
2.2.1 Códigos Maliciosos	6
2.2.1.1 <i>Worm</i>	6
2.2.1.2 <i>Spyware</i>	6
2.2.1.3 Cavalo de tróia	7
2.2.1.4 <i>Rootkit</i>	7
2.2.2 Ataques de Força Bruta	8
2.2.3 <i>SQL Injection</i>	8
2.2.4 Negação de Serviço DoS	8
2.3 Contramedidas	8
2.4 OWASP 2021	10
2.4.1 A01:2021 Quebra de Controle de Acessos (<i>Broken Access Control</i>).	12
2.4.2 A02:2021 Falhas de Criptografias (<i>Cryptographic Failures</i>).	12
2.4.3 A03:2021 Injeção (<i>Injection</i>).	12
2.4.4 A04:2021 Design Inseguro (<i>Insecure Design</i>).	13
2.4.5 A05:2021 Configuração Incorreta de Segurança (<i>Security Misconfiguration</i>).	13
2.4.6 A06:2021 Componentes vulneráveis e desatualizados (<i>Vulnerable and Outdated Components</i>).	15
2.4.7 A07:2021 falhas de identificação e autenticação (<i>Identification and Authentication Failures</i>).	15
2.4.8 A08:2021 Falhas de integridade de software e dados (<i>Software and Data Integrity Failures</i>).	16
2.4.9 A09:2021 As falhas de registro e monitoramento de segurança (<i>Security Logging and Monitoring Failures</i>).	16
2.4.10 A10:2021 Falsificação de solicitação via o lado do servidor (<i>Server-Side Request Forgery (SSRF)</i>).	17
3 – REVISÃO DA LITERATURA	18

4 – METODOLOGIA	24
5 – IMPLEMENTAÇÃO	26
5.1 OwaspZap 2.11.1	26
5.2 Quebra de autenticações.	29
5.3 Ataque de injeção utilizando BurpSuite	39
5.4 Implementação de Contramedidas	43
6 – CONCLUSÃO	44
6.1 Análise dos resultados	44
6.2 Trabalhos futuros	44
Referências	45

1 INTRODUÇÃO

Com o avanço da tecnologia da informação diversos problemas com segurança em aplicações *web* cresce junto. Todas as organizações armazenam em seus servidores grandes quantidades de informações dos clientes, fornecedores, dentre outros, informações essas que são fundamentais para o crescimento e competitividade no mercado.

De acordo com (MARCONDES, 2015) a segurança da informação tem o intuito de preservar a confidencialidade, integridade, disponibilidade, autenticidade e não repúdio da informação considerada importante para a organização. Com a grande quantidade de informações que circulam na rede tanto interna como externa, grandes problemas relativos à segurança também andam juntos. Nos últimos anos incidentes com vazamento de dados e até mesmo invasão tiveram um aumento significativo, com isso diversas formas de ataque vão sendo criadas e articuladas por invasores, ataques esses que são, pela rede sem fio e cabeada, acesso não autorizado em sistemas corporativos, implantação de *softwares* capaz de realizar negação de serviço, infecção de vírus, *spams*, *ransomware*, dentre muitas outras formas de ataque.

Políticas de Segurança, *firewall* bem configurados, técnicas de criptografia, sistemas atualizados, dentre outros mecanismos, ajudam as empresas a garantir a proteção de suas informação contra *cyber* ataques, que por sua vez reflete o comprometimento da empresa com a segurança.

Para definir boas práticas de segurança pode ter por base que todos os usuários da rede não são confiáveis, pois cada um tem uma visão diferente a respeito da segurança, muitos tentam burlar ou contornar a segurança, com isso deve se ter nível de acesso para limitar o que pode e o que não pode ser acessado, níveis esses que são definidos pelo administrador ou gerente de TI.

As falhas de segurança são um dos principais motivos de preocupação para empresas e usuários comuns, em que muitos desconhecem o risco de suas informações pessoais acabarem sendo expostas.

As preocupações das empresas relacionadas a uma ameaça aos negócios incluem os seguintes aspectos: tirá-las do cenário competitivo, ou prejudicar sua imagem, causar perdas financeiras, danificar outros ativos técnicos da empresa ou qualquer fator que afete diretamente sua segurança (DANTAS, 2011).

Nas organizações, as informações e redes estão cada vez mais sujeitos a fraudes digitais, espionagem, sabotagem, vandalismo, informações essas danificadas por códigos maliciosos, *hackers* e ataques de negação de serviço (SOUZA ADSON DIEGO LACERDA CAVALCANTE; SAMPAIO, 2018).

Os vírus de computador por exemplo, são códigos maliciosos com a função de se replicar e causar sérios danos ao computador como lentidão e abrir portas para que *hackers*

instalem *worms*, cavalo de troia e muitos outros, causando sérios problemas com o sigilo de informações, ficando expostas e derrubando todos os pilares da segurança da informação. Os *hackers* por sua vez estão usando métodos de ataque para roubar informação que estão causando transtornos para empresas como, por exemplo, o *CryptoLocker*, é um *ransomware* que sequestra dados importantes, que criptografa todos os dados de um servidor ou sistema e cobram uma fortuna pelo resgate (LIMA, 2016).

Com adoção de medidas de segurança é possível se prevenir contra ataques de diversos tipos, dos quais podemos citar: manter *software* de antivírus, *antispyware* sempre atualizados; utilizar *firewall* na rede que quando devidamente configurados ajuda a impedir invasão e impede que envie programas maliciosos para outros computadores na rede. Tomar todos os cuidados preventivos aumentam consideravelmente o nível de proteção das informações, realizar *backups* sempre que possível, é um meio adequando para salvaguardar um possível ataque.

Segundo (NOLETO, 2020) uma aplicação *web* refere-se a uma solução que roda diretamente em um navegador sem precisar ser instalada no computador do usuário. Pode ser definida também como algo processado em um servidor terceiro. Por exemplo, plataformas de e-commerce e redes sociais se encaixam nesse perfil. Gerenciar um aplicativo da *web* não é apenas sobre servidor e hospedagem, para evitar riscos a segurança, deve se atualizar constantemente métodos de segurança deste aplicativo, que ajuda a reduzir riscos de ataque.

Diante disso uma entidade sem fins lucrativos denominada *OWASP* (*Open Web Application Security Project*) reconhecida internacionalmente, atua com o objetivo de realizar pesquisas e levantamentos sobre diversas categorias de vulnerabilidades que mais impactam as aplicações *web*. De quatro em quatro anos o *OWASP* atualiza a lista do top 10 das vulnerabilidades *web* que foi o documento utilizado como base para este projeto, para analisar as principais vulnerabilidades e apresentar contramedidas para auxiliar futuros desenvolvedores.

1.1 Objetivos

Diante do exposto, este projeto tem o objetivo geral de analisar as principais vulnerabilidades de segurança em aplicações *web*, seus riscos e contramedidas.

1.1.1 Objetivos Específicos

- Apresentar os diversos tipos de ataque recorrentes em aplicações *web*.
- Enunciar mecanismos de segurança utilizada para conter ataques.
- Divulgar e popularizar o *OWASP*, bem como suas recomendações para proteger aplicações *web*.

- Contribuir com a comunidade acadêmica e sociedade em geral, no sentido de prevenir ataques a aplicações *web*.

1.2 Organização do Trabalho

A estrutura deste trabalho é composta por seis seções onde serão abordados assuntos sobre as principais ameaças de acordo com *OWASP*, definições de conceitos sobre segurança da informação. Em resumo, pode-se definir os capítulos seguintes em:

- Capítulo 1 – Introdução: É abordado uma introdução sobre avanço da tecnologia; fundamentos da confidencialidade, integridade, disponibilidade, autenticidade e não repúdio.
- Capítulo 2 – Fundamentação Teórica: É abordado fundamentos do *OWASP*; principais ataques de acordo com o *OWASP*; Códigos Maliciosos; Metodologia de classificação de risco.
- Capítulo 3 – Revisão da Literatura: Onde são elencadas diversificadas referências e o estado da arte relativo à segurança da informação.
- Capítulo 4 – Metodologia: Metodologia utilizada na elaboração deste trabalho.
- Capítulo 5 – Implementação: Onde serão mostrados laboratórios utilizando, ferramentas de análise e exploração de vulnerabilidades em aplicações *web*.
- Capítulo 6 – Conclusão: Onde é definido a conclusão deste trabalho bem como a análise dos resultados e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

A seguir, serão apresentados os conceitos utilizados neste trabalho.

2.1 Segurança da Informação

Na atualidade estamos rodeados de informações espalhadas em diversos meios tecnológicos, através delas podemos tomar decisões e até mesmo ideias de novos negócios, o acesso a essas informações muitas vezes visíveis a qualquer momento e lugar, com isso deve se pensar bem a respeito da segurança e resguardo dessas informações.

Oportunidades de negócios e rápidos avanços no mundo globalizado vem e vão com muita rapidez, vivenciamos grandes transformações tecnológicas, econômica e mercadológicas. Devido a grandes fusões de infraestrutura de telecomunicações acarreta sérios problemas de segurança. Embora haja diversas variedades de ferramentas de segurança e técnicas de seguranças, administradores de segurança passam por dificuldades quanto as ações no âmbito de proteger um rede ou um sistema.

Levando em consideração as novas tecnologias, os processos de negócios e o pessoal continuarão mudando, alterando assim o nível atual de risco e podem criar novas ameaças para a organização. Com isso as medidas de controle relacionadas à segurança da informação devem ser estrita e metodicamente avaliadas (WESTPHALL; RIGON, 2013).

A segurança da informação é um conjunto de medidas que são usadas para garantir os cinco pilares que sustentam a segurança de TI (Tecnologia da Informação) que são eles: confidencialidade, integridade, disponibilidade, autenticidade e não repúdio (BENETTI, 2015).

- Confidencialidade tem por objetivo preservar a privacidade de dados de uma organização, assegura-se que dados não sejam conhecidos por pessoas que não sejam autorizadas (BENETTI, 2015).

- Integridade visa garantir que dados sejam íntegros desde o remetente até o destinatário, assegurando que o mesmo não seja modificado (BENETTI, 2015).

- Disponibilidade garante que toda informação possa ser obtida sempre que precisar (BENETTI, 2015).

- Autenticidade em segurança de dados significa que o remetente de uma mensagem, o autor de um documento ou o operador de uma transação é realmente quem afirma ser (VERAS, 2018).

- Não repúdio é provar quem foi o autor de determinada ação, e utilizam métodos como criptografia que impedem que individuos neguem a execução de uma ação (VERAS, 2018).

2.2 Aplicações Web e suas Vulnerabilidades

Segundo (DURBANO, 2019) o *CVE*¹, ou *Common Vulnerabilities and Exposures*, é um projeto colaborativo de várias organizações de tecnologia e segurança que cria listas de nomes padrão para vulnerabilidades e outros riscos de segurança da informação. O *CVE* visa padronizar vulnerabilidades e riscos conhecidos, tornando mais fácil encontrar, acessar e compartilhar informações entre diferentes indivíduos e empresas.

Mais do que uma lista, o *CVE* é uma espécie de dicionário de vulnerabilidades do mundo virtual. Esta ferramenta é mantida por representantes de organizações de segurança, instituições acadêmicas, governos e diversos especialistas.

De acordo com o (DETAILS, 2022) a Figura 1 elenca as vulnerabilidades e o número que vezes que foram detectadas desde 1999 a 2022.

Figura 1 – Vulnerabilidades CVE em anos

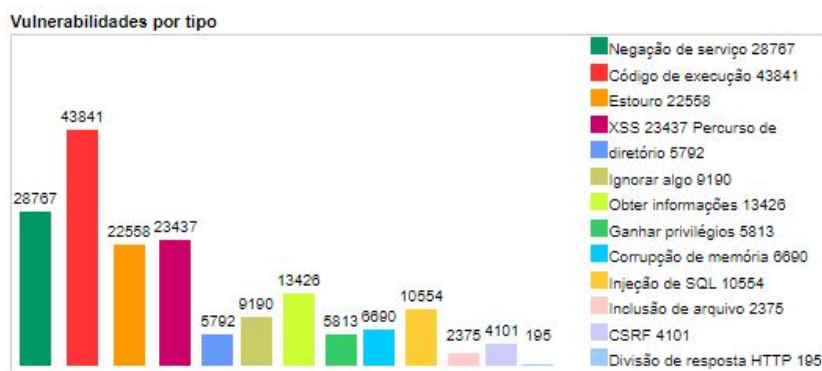
Ano	Nº de vulnerabilidades	DoS	Execução de código	Transbordar	Corrupção de Memória	Injeção SQL	XSS	Percurso de diretório	Divisão de resposta HTTP	Ignorar algo	Informações de ganho	Ganho de privilégios	CSRF	Inclusão de arquivo	# de explorações
1999	894	177	112	172			2	7		25	16	103			2
2000	1020	257	208	206	1	2	4	20		48	19	139			
2001	1677	403	403	297		7	34	124		82	36	220		2	2
2002	2156	498	353	433	2	41	200	103		127	76	199	2	14	1
2003	1527	381	477	372	2	50	129	60	1	62	69	144		16	3
2004	2451	580	614	408	3	148	291	111	12	145	96	134	5	38	5
2005	4935	838	1627	657	21	604	786	202	15	289	261	221	11	100	14
2006	6610	893	2719	666	91	967	1302	322	8	267	272	184	18	849	30
2007	6520	1101	2601	954	95	706	883	338	14	267	326	242	69	700	45
2008	5632	894	2310	699	128	1101	807	362	7	288	268	188	83	170	76
2009	5736	1035	2185	698	188	963	832	323	9	337	302	223	115	138	738
2010	4653	1102	1714	672	342	520	605	276	8	234	284	238	86	73	1501
2011	4155	1221	1334	723	351	294	470	109	7	197	408	206	58	17	557
2012	5297	1423	1459	832	423	243	739	122	13	344	391	250	166	14	623
2013	5191	1455	1186	850	366	153	650	110	7	352	511	274	123	1	206
2014	7939	1599	1572	841	420	304	1103	204	12	457	2107	239	264	2	403
2015	6504	1793	1830	1082	749	221	784	151	12	577	752	266	248	5	128
2016	6454	2028	1496	1311	717	94	498	99	13	444	867	501	86	7	1
2017	14714	3137	3004	2486	743	308	1518	279	11	629	1648	459	327	18	6
2018	16557	1855	3041	2124	400	517	2048	545	11	708	1231	247	461	31	4
2019	17344	1344	3201	1243	488	552	2391	476	10	712	914	202	535	57	13
2020	18325	1351	3249	1533	409	464	2182	415	14	966	1215	310	402	37	62
2021	20171	1838	3850	1673	484	740	2710	526	5	880	807	260	505	46	
2022	18566	1542	3096	1622	265	1353	2429	508	4	752	550	164	537	40	
Total	185028	28767	43841	22558	6690	10554	23437	5792	195	9190	13426	5812	4101	2375	4423
% De tudo		15.5	23.7	12.2	3.6	5.7	12.7	3.1	0.1	5.0	7.3	3.1	2.2	1.3	

Fonte: (DETAILS, 2022)

A Figura 2 é um gráfico comparativo para melhor visualização da quantidade de vezes que uma determinada vulnerabilidade foi catalogada desde 1999 a 2022.

¹<https://www.cvedetails.com/vulnerabilities-by-types.php>

Figura 2 – Principais tipos de vulnerabilidades



Fonte: (DETAILS, 2022)

2.2.1 Códigos Maliciosos

Códigos maliciosos são programas desenvolvidos especialmente para executar nos computadores tarefas maliciosas que por sua vez tem ação danosa, quando instalados eles passam ter acesso a dados armazenados no computador, e executam ações sem a permissão do usuário.

O Objetivo do atacante é desenvolver e espalhar códigos maliciosos, obtenção de informações confidenciais de usuários, apropriação de informação e bens financeiros. Muitos atacantes usam esse meio para propagar vandalismo e utilizam os códigos como intermediários para disseminar *spam* (SEGURANÇA, 2017).

2.2.1.1 Worm

Worm é um programa capaz de se propagar pela rede enviando cópias de si mesmo de computador para computador, quando suas cópias são executadas diretamente. Uma das funções do *worm* é consumir recursos do computador fato esse que ocorre quando suas cópias são criadas e propagadas.

O processo de propagação e infecção dos *worms* ocorre da seguinte maneira: Primeiramente ele faz a identificação dos computadores ativos na rede para serem alvos, logo após, ele começa a fazer o envio das cópias por meio de anexos de email, programas de troca de mensagens, de pastas compartilhadas. Feito isso ele começa a ativação de suas cópias realizando a execução diretamente pelo usuário e até mesmo pela inserção de mídias removíveis. E, por último, o computador que era um alvo passa a ser um originador dos ataques e o processo todo reinicia (SEGURANÇA, 2017).

2.2.1.2 Spyware

Spyware foi projetado para monitorar atividades dos sistemas e enviar informação para o atacante. O *Spyware* pode ser instalado de forma legítima quando o próprio dono do

computador permite a instalação, para obter dados de acesso e com o objetivo de verificar se outra pessoa utiliza o computador de forma não autorizada. Pode ser instalado também de forma maliciosa quando o objetivo é ter acesso a informação de navegação como por exemplo nomes de usuário e senhas.

Podemos citar alguns programas *spywares* dentre eles estão o *keylogger* que cria logs de tudo que é digitado no computador. O *screenlogger* que armazena posição do mouse quando clicado, utilizado para capturar senhas de teclados virtuais disponíveis em sites de *Internet Banking*, e por último o *adware* que tem a função de apresentar propagandas em *sites*, muitas propagandas com intuito financeiro ou seja para direcionar para algum site de compra, e também com intuito malicioso quando as propagandas apresentadas são direcionadas, de acordo com a navegação do usuário e sem que este saiba que tal monitoramento está sendo feito (SEGURANÇA, 2017).

2.2.1.3 Cavalo do tróia

Segundo a mitologia grega, o "Cavalo de tróia" foi uma grande estátua que os gregos utilizaram como arma de guerra para entrar na cidade de Tróia. Os soldados ficaram dentro da estátua do cavalo e abriram os portões à noite para permitir que os gregos entrassem e invadissem Tróia.

No computador ele tem função semelhante e se passa por um programa aparentemente inofensivo para que outras pessoas baixem e instalem no computador e consequentemente ele faz mudança no sistema de segurança para que outros *malwares* e até mesmo cibercriminosos possam invadir (REGAN, 2019).

O autor cita alguns "famosos cavalos de Troia" que se tornaram bem conhecidos são eles: *PC-Write*, *AIDS*, *Back Orifice*, *Pretty Park*, *NetBus*, *SubSeven*, *BO2K* e *Zeus*. Eles possuem diversas características em comum, o mais antigo o *PC-Write* lançado em 1986 ele apagava a tabela de alocação de arquivos e formatava o disco rígido, já o mais recente o *Zeus* lançado em 2007 está associado a *botnets* em 2010 esteve envolvido em ataques a várias agências governamentais, sendo que a principal função dele é ter acesso a informações financeiras (BASTA; BROWN, 2014).

2.2.1.4 Rootkit

Segundo o site *Canaltech*, o *rootkit* é um *software* criado para esconder e camuflar certos processos ou programas, ele funciona como um porteiro (*backdoor*) e permite que um usuário bem ou mal intencionado circule livremente sem que mecanismo de segurança inibem tais ações (REDACAO-CANALTECH, 2016).

2.2.2 Ataques de Força Bruta

Um ataque de força bruta consiste em tentar decifrar uma senha ou nome de usuário, encontrar a página da *Web* oculta ou a chave usada para criptografar uma mensagem por tentativa e erro e esperar que em algum momento ela possa ser adivinhada. Este é um método de ataque antigo, mas ainda popular e eficaz entre os atacantes. (KASPERSKY, 2022).

Segundo (MOLINARO, 2021) Na década de 1970, um atacante poderia, teoricamente, tentar apenas milhares de variações de senhas diferentes a cada segundo. Hoje, a computação moderna permite centenas de bilhões de tentativas de *login* por segundo. Embora o significado de força bruta tenha sido expandido, o método permanece o mesmo, tentar o maior número possível de combinações de senhas e esperar pela correta.

2.2.3 SQL Injection

A injeção de *SQL* (*Structured Query Language*) é uma vulnerabilidade de segurança de rede que permite que um invasor intercepte consultas feitas por um aplicativo em um banco de dados. Isso geralmente permite que um invasor veja informações que normalmente não conseguiria recuperar. Isso pode incluir informações pertencentes a outros usuários ou outras informações que o próprio aplicativo pode acessar. Em muitos casos, um invasor pode modificar ou excluir essas informações, causando alterações permanentes no conteúdo ou na operação do aplicativo. Em algumas situações, um invasor pode escalar um ataque de injeção de *SQL* para comprometer o servidor subjacente ou outra infraestrutura de *back-end* ou realizar um ataque de negação de serviço (PORTSWIGGER, 2021).

2.2.4 Negação de Serviço DoS

De acordo com o site (CLOUDFLARE, 2022) Um ataque de negação de serviço (DoS) é um tipo de ataque destinado a tornar um computador, aplicação *web* ou outro dispositivo inacessível ao usuário, interferindo assim no funcionamento normal. Os ataques DoS geralmente funcionam sobrecarregando o computador de destino com solicitações até que ele não consiga mais lidar com o tráfego normal, resultando em uma negação de serviço.

2.3 Contramedidas

Segundo o site (KASPERSKY, 2022) para mitigar os ataques a força bruta em aplicações *web* os administradores do sistema devem garantir que as senhas do sistema sejam criptografadas usando a taxa de criptografia mais alta possível. Quanto mais *bits* no esquema de criptografia, mais difícil é adivinhar a senha.

Os administradores também devem executar um procedimento chamado *salt hash*, que torna aleatórios os *hashes* de senha adicionando uma sequência aleatória de letras

e números (chamada *salt*) à própria senha. Essa sequência deve ser armazenada em um banco de dados separado e recuperada e adicionada à senha antes do *hash*. Portanto, usuários com a mesma senha confiam em *hashes* diferentes. Além disso, os administradores podem exigir autenticação de dois fatores e instalar um sistema de detecção de intrusão que possa detectar ataques de força bruta.(KASPERSKY, 2022)

Limitar o número de tentativas também reduz a vulnerabilidade a ataques de força bruta. Por exemplo, permitir que um usuário digite a senha correta três vezes antes de bloqueá-lo por alguns minutos pode introduzir atrasos significativos e incentivar os *hackers* a procurar alvos mais fáceis.(KASPERSKY, 2022)

De acordo com (MACORATTI, 2002) para minimizar ataques de *SQL injection* deve se:

- Criar políticas de segurança rigorosa e criteriosa que limite o acesso de seus usuários. Isso significa que deve se dar a usuários apenas as permissões de que precisam. Não fornecer acesso de gravação às tabelas e o acesso é concedido apenas às tabelas que o usuário precisa.
- Valide as informações inseridas no formulário e não permita caracteres inválidos como: (') , (- -) e (;) e palavras nocivas como *insert* , *drop* , *delete* , *xp_* .
- Não mostrar o nome de usuário caso somente a senha estiver errada. Exemplo: Quando se faz *login* em um *site*, o atacante digita o nome de usuário e senha, o atacante acerta o nome de usuário e erra a senha, o *site* mostra a mensagem que a senha é inválida para o usuário digitado, ou seja nesse caso o ataque já conhece o nome do usuário, podendo assim injetar um código para ignorar a senha.

Com base em todas as recomendações do *OWASP* para as vulnerabilidades existentes mais importantes, segue abaixo um guia de boas práticas e manutenção de uma aplicação *web* mais segura para que os desenvolvedores possam tornar a Internet mais segura e tentar evitar que sistemas sejam comprometidos e *hackers* acessem informações confidenciais.

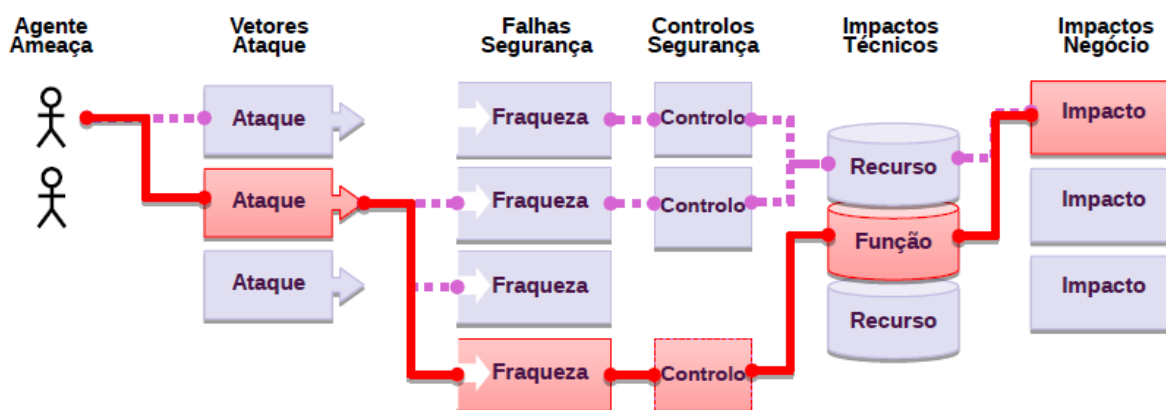
- A entrada de usuários devem ser sanitizadas ou seja com base nos termos ou caracteres permitidos, e dados inválidos devem ser removidos antes do processamento.
- Utilizar políticas de senha forte para usuários e administradores.
- Bloquear ips automaticamente após um certo número de tentativas de *login* inválidas.
- Implementar autenticação de múltiplos fatores a qual um aplicativo ou um sistema confirma sua identidade em dois ou mais momentos, antes de liberar o acesso ao sistema.
- Sempre manter atualizados os *softwares*, *CMS* (*Content management system*), *plug-ins* e todos os componentes que compõem a aplicação.

2.4 OWASP 2021

Segundo relatório de 2021 da *OWASP* (2021), que é uma comunidade aberta sem fins lucrativos que trabalha pra melhorar a segurança de *software* e APIs (*Application Programming Interface*) confiáveis, relata que *softwares* inseguros podem afetar nossas finanças, saúde, defesa, energia e outras infraestruturas. À medida que o programa se torna crítico, complexo e interconectado, a dificuldade de garantir a segurança da aplicação cresce exponencialmente (OWASP, 2021).

A cada ano que passa os riscos à segurança da informação aumentam de forma exponencial, atacantes utilizam caminhos diversos pela aplicação para afetar um negócio e até mesmo uma organização. A Figura 3 elucida o caminho mais utilizado por um atacante até chegar ao ponto de proporcionar na organização um determinado impacto, caminhos estes que podem ser difíceis de percorrer, porém de forma que os danos causados podem não ter consequências, podendo mesmo causar danos a toda a organização.

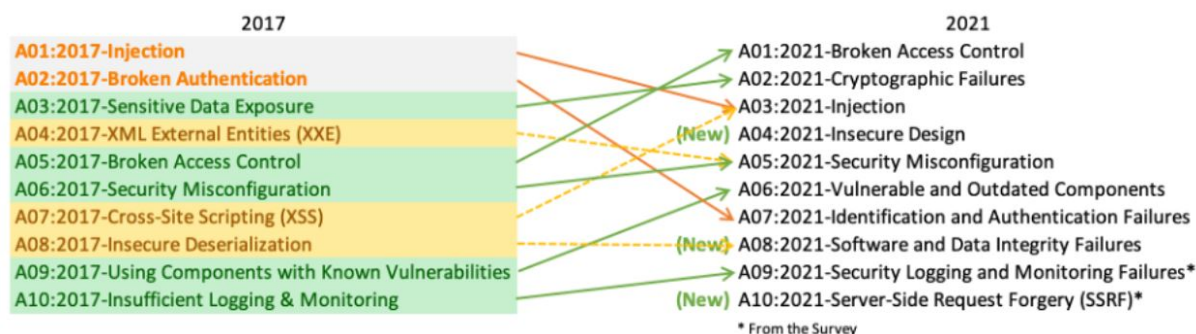
Figura 3 – Fluxograma utilizado pelo atacante para atingir uma aplicação



Fonte: Owasp (2021)

Desde 2017, quando foi publicada a última lista das 10 maiores vulnerabilidades, uma análise da classificação de risco realizada pelo *OWASP* no ano de 2017 em relação com a última publicação em 2021 revela novas tendências de vulnerabilidades de segurança no ciclo de desenvolvimento *web*.

Figura 4 – Comparação do Owasp de 2017 para 2021



Fonte: (OWASP, 2021)

Uma análise comparativa na Figura 4 mostra que todas as categorias foram mantidas de 2017 a 2021, mas a classificação mudou. Em geral, algumas categorias de edições anteriores foram mescladas e combinadas em uma categoria. Porque, de acordo com (OWASP, 2021), essas mudanças devem se concentrar nas causas raiz e não nos sintomas.

Também foram adicionadas três novas categorias, que são: A04: 2021-*Insecure Design*, que foca nos riscos associados ao design de aplicativos e falhas de arquitetura; A08:2021- *Software and Data Integrity Failures* que faz suposições sobre atualizações de *softwares* e A10:2021-*Server-Side Request Forgery (SSRF)*, que foi a ultima categoria adicionada ao *OWASP* (OWASP, 2021).

De acordo com a análise do site (HKCERT, 2021), as mudanças do *OWASP* 2017 em relação ao *OWASP* 2021 foram resumidas da seguinte forma:

- A exposição de dados sensíveis (*Sensitive Data Exposure*) agora é reclassificada como uma falha de criptografia (*Cryptographic Failures*) porque esse tipo de falha é, na verdade, a causa raiz de exposição de dados confidenciais.
- *Cross-Site Scripting (XSS)* está incluído na categoria *Injection*, que é tecnicamente a inserção de um *script* que tem como alvo outros usuários que visitarem o site. Este é um ataque diferente dos métodos de injeção do lado do servidor, como *SQL*, comandos do SO (Sistema Operacional) e injeções de *LDAP (Lightweight Directory Access Protocol)*.
- A vulnerabilidade *XML External Entities (XXE)* agora foi reclassificada como Configuração Incorreta de Segurança (*Security Misconfiguration*) porque é um erro comum cometido por desenvolvedores que configuram incorretamente analisadores XML em seu código.
- A desserialização insegura (*Insecure Deserialization*) é colocada em falhas de integridade de software e dados (*Software and Data Integrity Failures*). Como esse é um problema de segurança insuficiente ao lidar com objetos serializados, isso leva à alteração de dados e à transferência desses dados para aplicativos de fontes não confiáveis. Um número significativo de desserializações inseguras se deve a vulnerabilidades de código aberto que devem ser gerenciadas ativamente e não ignoradas.

2.4.1 A01:2021 Quebra de Controle de Acessos (*Broken Access Control*).

As restrições às operações permitidas a usuários autenticados nem sempre podem ser verificadas corretamente. Os invasores podem usar essas falhas para ativar funções ou dados não autorizados, como dados de outras contas de usuário, visualizar arquivos confidenciais, modificar dados de outros usuários, alterar permissões de acesso, dentre outros (OWASP, 2021).

O acesso não autorizado à rede ou aos serviços de rede pode causar sérias consequências comerciais. Portanto, além de gerenciar adequadamente os direitos de acesso à rede pelos usuários, as organizações também devem estar cientes de suas responsabilidades (MACHADO, 2012).

2.4.2 A02:2021 Falhas de Criptografias (*Cryptographic Failures*).

Muitos aplicativos da *web* não protegem adequadamente dados confidenciais, como cartões de crédito, identificações fiscais e credenciais de autenticação. Os invasores podem roubar ou modificar esses dados desprotegidos para cometer fraude de cartão de crédito, roubo de identidade ou outros crimes. Os dados confidenciais devem receber proteção adicional, como criptografia durante o armazenamento ou transmissão, e precauções especiais quando o navegador é usado para gerenciamento de tráfego (OWASP, 2021).

Outro exemplo de exposição são os dispositivos na rede, que possuem uma tabela ARP (*Address Resolution Protocol*) que armazena qual é o *IP* (*Internet Protocol*) que possui um determinado endereço *MAC* (*Media Access Control*) na rede. Para fazer isso, frequentemente as máquinas da rede enviam requisições para atualizar sua tabela, é possível configurar um endereço para enganar outras máquinas e forçarem a atualização da tabela para identificar o seu *MAC* como o *IP* de outra máquina (SANDERS, 2010).

2.4.3 A03:2021 Injeção (*Injection*).

Quando dados não confiáveis são enviados ao intérprete como parte de um comando ou consulta, podem ocorrer falhas de injeção, como injeção de *SQL*, *SO* e *LDAP*. Os dados manipulados pelo invasor podem induzir o intérprete a executar comandos indesejados ou permitir o acesso a dados não autorizados. Muitas fontes de dados podem sofrer ataque de injeção, tanto serviço *web* interno e externo, sistemas locais que utilizam um banco de dados e que possuem falhas na segurança estão sujeitos a sofrerem ataques de injeção (OWASP, 2021).

De acordo com o *OSWAP* falhas relacionadas à injeção são muito comuns, especialmente em códigos antigos. Eles são normalmente encontrados em *SQL*, *LDAP*, *XPath* (*XML Path Language*), comandos do sistema, processadores XML (*Extensible Markup Language*), cabeçalhos *SMTP* (*Simple Mail Transfer Protocol*), linguagem de expressão e consultas *ORM* (*Object-relational mapping*). É fácil encontrar esses defeitos analisando o

código. *Scanners e fuzzers* podem ajudar o invasor a descobrir falhas de injeção (OWASP, 2021).

O ataque de injeção pode causar um impacto grande em uma organização como perda de dados, corrupção de dados, falhas de negação de acesso e pode levar a perda do controle do sistema. A revisão do código da aplicação é uma das formas de se prevenir contra um ataque de injeção, a realização de testes automáticos que englobam cabeçalhos, *URL (Uniform Resource Locator)*, *cookies*, *JSON (JavaScript Object Notation)*, e dados de entrada para *XML* (OWASP, 2021).

No OWASP 2017 a falha de XSS ocupava a 7^o posição da lista, na lista do OWASP de 2021 foi realizada a junção juntamente com a falha de injeção que ocupa a 3^a posição do *ranking*.

A falha de XSS ocorre sempre que um aplicativo recebe dados não confiáveis e os envia para o navegador sem verificação ou filtragem adequada. XSS permite aos invasores executarem *scripts* no navegador da vítima que podem 'sequestrar' sessões do usuário, desfigurar *sites*, ou redirecionar o usuário para sites maliciosos (OWASP, 2021).

2.4.4 A04:2021 Design Inseguro (*Insecure Design*).

De acordo com (OWASP, 2021), *Insecure Design* é uma nova categoria para 2021 que concentra-se nos riscos associados aos erros de *design* e arquitetura, com o objetivo de alertar os desenvolvedores para fazer mais uso de modelagem de ameaças, padrões de *design* seguros e arquiteturas de referência para prevenir ataques e implementar medidas proativas para proteger adequadamente aplicações *web*.

Com base em dados de teste fornecidos pelo (OWASP, 2021), a fundação diz que o *design* inseguro não deve ser considerado a fonte de todas as outras 10 categorias de risco principais porque há uma diferença entre *design* inseguro e implementação insegura.

(HKCERT, 2021) menciona que a lógica por trás das ferramentas automatizadas pode criar riscos de segurança para as empresas que as utilizam se não forem devidamente monitoradas. Um exemplo disso é a violação de dados no Facebook no início de 2021. Segundo (RIGUES, 2021), das 533 milhões violações de dados no *facebook* foram causadas por agentes maliciosos que adulteraram a ferramenta de importação de contatos para obter uma quantidade limitada de informações de perfil. A principal causa dessa violação foi um uma arquitetura insegura aplicada no uso de uma ferramenta automatizada. Um design de aplicação mal elaborado pode trazer graves consequências para um negócio.

2.4.5 A05:2021 Configuração Incorreta de Segurança (*Security Misconfiguration*).

De acordo com (OWASP, 2021) subiu da 6^o posição na edição anterior para a 5^o posição na edição de 2021: 90% dos aplicativos foram testados para algum tipo de erro com incidência média de 4,5% e mais de 208 mil eventos *CWE (Common Weakness Enumeration)* foram mapeados para essa categoria de risco. À medida que mais alterações

são feitas em software altamente personalizável, não é surpresa que essa categoria esteja em ascensão. A antiga categoria de Entidades Externas *XML* agora faz parte desta categoria de risco.

De acordo com (OWASP, 2021) muitos processadores de *XML* antigos ou mal configurados fazem referências a entidades externas dentro dos documentos *XML*. Essas entidades externas podem ser usadas para revelar arquivos internos usando o processador de *URI* (*Uniform Resource Identifier*), compartilhamento interno de pastas, pesquisa de portas de comunicação internas, execução de código remoto e ataques de negação de serviço, tal como o ataque *Billion Laughs* que é o ataque de um bilhão de risos é um tipo de ataque de negação de serviço que visa analisadores de documentos *XML*. Algumas ferramentas podem detectar este problema realizando análises de dependências que consiste em obter uma estrutura sintática correspondente a determinado texto da linguagem natural. O ataque de negação de serviço *DoS* é uma estratégia concebida para evita que o tráfego válido chegue ao alvo de ataque que são servidores web, servidores de aplicativos e *links* de comunicação, o ataque *DoS* (*Disk Operating System*) nega acesso ao tráfego de rede legítimo interrompendo e danificando conexões entre computadores, impedindo que usuários finais acessem serviços de uma máquina, e em alguns casos destrói componentes da rede física, e podem acarretar graves problemas como destruir dados e levar a perdas financeiras bastante significativas. Um exemplo prático de ataque de negação por exemplo se um servidor *web* só e capaz de aceita mil pedidos por segundo, então o 1001º será negado.

O autor cita algumas ferramentas e suas principais características usadas para ataques *DDoS* (*Distributed Denial of Service*) são elas: *Trinoo*, *Stacheldraht* e *botnets*, um ataque via rede *Trinoo* o invasor inicia ataques contra um ou mais endereços de *IP*, logo o servidor comunica-se com seus clientes, dando instruções para atacar seus endereços em determinando período de tempo. O *Stacheldraht* e bem semelhante ao *Trinoo* os computadores manipuladores são gerenciados por clientes *criptografados*, um invasor e capaz de lidar com vários manipuladores ao mesmo tempo aumentando assim o numero de sistemas atingidos. Os *Botnets* mencionados anteriormente são instalados por *backdoor* podendo ser controlado por um invasor, pela rede *botnet* aproveita seu poder para causar danos consideráveis ou simplesmente enviar grandes quantidades de lixo eletrônico (BASTA; BROWN, 2014).

Segundo o (OWASP, 2021) um invasor pode usar um aplicativo que processa *XML* confiável para se conectar a outros sistemas internos, possivelmente divulgando outro conteúdo interno por meio de solicitações *http(s)* (*Hyper Text Transfer Protocol Secure*) ou lançando um ataque a quaisquer serviços internos desprotegidos. O padrão *XML* 1.0 define um conceito denominado entidade, que é uma unidade de armazenamento. Existem vários tipos diferentes de entidades. Entidades externas gerais analisadas por parâmetros, geralmente chamadas de entidades externas, podem acessar conteúdo local ou remoto por meio de identificadores de sistema declarados. O identificador do sistema é considerado

um URI e o processador *XML* pode acessar o *URI* ao processar a entidade. Em seguida, o processador *XML* substitui a aparência da entidade externa nomeada pelo conteúdo não cotado com o identificador do sistema.(OWASP, 2021).

Para se ter uma aplicação segura, requer configurações de segurança implementadas e definidas no aplicativo, na estrutura, no servidor, no servidor da *web* e no banco de dados. Todas essas configurações devem ser definidas, implementadas e mantidas, já que geralmente a configuração padrão é insegura(OWASP, 2021).

2.4.6 A06:2021 Componentes vulneráveis e desatualizados (*Vulnerable and Outdated Components*).

Componentes vulneráveis e desatualizados, anteriormente chamados de "Usando componentes com vulnerabilidades conhecidas", passaram do 9º lugar e agora estão em 6º na lista. Ele tinha informações suficientes para entrar no Top 10 pela análise de dados. Esta categoria pode ser difícil de testar e avaliar o risco, e é a única categoria que não possui *CVEs* com *CWEs* incluídos. Toda organização deve ter um plano contínuo para rastrear, solicitar e implantar atualizações ou alterações de configuração ao longo do ciclo de vida de um aplicativo (OWASP, 2021).

Este tipo de vulnerabilidade pode acontecer ao usar componentes com vulnerabilidades conhecidas. A maioria desses componentes são bibliotecas, *frameworks* ou módulos com privilégios elevados. Em um ataque, esses componentes podem ser usados para causar perda de dados ou danos ao servidor (OWASP, 2021).

2.4.7 A07:2021 falhas de identificação e autenticação (*Identification and Authentication Failures*).

Anteriormente na edição de 2017 do *OWASP* essa falha ocupava o 2º e recebia o nome de quebra de autenticação, na edição de 2021 do *Owasp* ela desceu para a 7º posição recebendo o nome de falhas de identificação e autenticação.

As funções do aplicativo relacionadas à autenticação e ao gerenciamento de sessão costumam ser implementadas incorretamente, permitindo que os invasores comprometam as senhas, chaves e *tokens* da sessão ou usem outra falha de implementação para assumir a identidade de outros usuários. De acordo com o *OSWAP*, um invasor pode acessar um grande número de combinações válidas de nome de usuário e senha para ataque de *credential stuffing* (teste exaustivo), ataques de força bruta e de dicionário e acesso a contas de administrador padrão. Os invasores podem usar funções e ferramentas automáticas (como listas de senhas e dicionários de contas) para remover quebras de autenticação (OWASP, 2021).

Quando o processo de expiração de sessão não está bem implementado no sistema ele pode ser considerado vulnerável, um usuário acessa um determinado sistema em um computador público e quando termina de utilizar ele simplesmente fecha o navegador sem

clicar em *logout* ou sair, o atacante utiliza o mesmo computador mais tarde e mesmo assim a sessão continua ativa e autenticada.

Para o atacante comprometer o sistema basta ter apenas a conta de usuário administrador, e pode levar a graves problemas como, roubo de identidade social, fraudes, lavagem de dinheiro juntamente com extorsão por parte do atacante, e exposição de informações confidenciais. Quando se tem um sistema que permite usuários criarem senhas fracas como “*admin/admin*”, “*123456*” e que não tem o método de autenticação de dois fatores, a possibilidade de *hackers* invadir e subtrair a identidade de usuários é grande.

2.4.8 A08:2021 Falhas de integridade de software e dados (*Software and Data Integrity Failures*).

A categoria Desserialização Insegura apresentada na 8^o posição na edição do TOP 10 de 2017 foi reconstituída nesta nova categoria, e segundo (OWASP, 2021), a nova categoria é focada em realizar suposições relacionadas à falta de verificação de integridade durante atualizações de *software*, dados críticos e *pipelines*.

De acordo com (HKCERT, 2021) em dezembro de 2020, a empresa de *software* de gerenciamento de rede *SolarWinds* foi invadida e os atacantes injetaram código malicioso em seu ambiente de desenvolvimento. Sem um processo de verificação de integridade de código eficaz, o código malicioso foi entregue a mais de 18.000 organizações, causando um dos ataques cibernéticos mais graves de todos os tempos. A assinatura de código com repositórios confiáveis e ferramentas de segurança da cadeia de suprimentos de *software* são precauções que podem ser implementadas para neutralizar essas ameaças.

(SOLARWINDS, 2022) é uma empresa que desenvolve soluções de *software* para gerenciamento de redes, sistemas, bancos de dados e segurança de *TI*. Segundo (TURNER, 2020) o Orion que é um dos *softwares* de gerenciamento de rede, desenvolvido pela *SolarWinds*, sofreu um ataque, a qual agentes mal-intencionado implantou um código malicioso no *software* e o adulterou antes que a atualização fosse lançada para todos os usuários, resultando na entrega de *malware* aos clientes da *SolarWinds*. Com isso, após um *malware* instalado, os atacantes por vezes conseguem obter controle total sobre as redes do cliente.

2.4.9 A09:2021 As falhas de registro e monitoramento de segurança (*Security Logging and Monitoring Failures*).

A vulnerabilidade: As falhas registro e o monitoramento de segurança vieram da pesquisa da comunidade Top 10 (n^o 3), ocupando a 10^o posição no OWASP Top 10 2017, sendo no OWASP top 10 2021 subiu para 9^o posição. O registro e o monitoramento podem ser difíceis de testar, geralmente envolvendo entrevistas ou perguntando se os ataques foram detectados durante uma penetração teste. Não há muitos dados *CVE* para essa categoria, mas detectar e responder a violações é fundamental. Ainda assim, pode ser

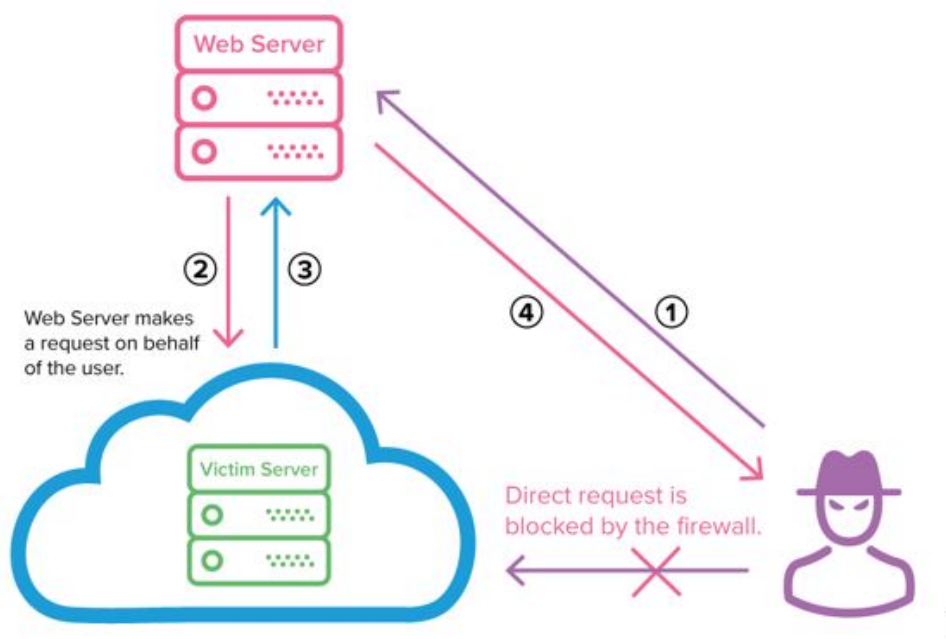
muito impactante para responsabilidade, visibilidade, alerta de incidentes e análise forense (OWASP, 2021).

Os autores do *OWASP* enfatizam a abrangência desse problema, o que significa que se os eventos da aplicação *web* e do servidor forem monitorados de forma inadequada, atividades suspeitas podem facilmente passar despercebidas e que um método de registro bem implementado alerta sempre que ocorrem anomalias, e o monitoramento cuidadoso permite uma ação mais rápida para evitar explorações de vulnerabilidades.

2.4.10 A10:2021 Falsificação de solicitação via o lado do servidor (*Server-Side Request Forgery (SSRF)*).

Como os aplicativos da *web* modernos fornecem recursos convenientes para usuários finais, a busca de *URL* está se tornando um cenário comum. Como resultado, a incidência de *SSRF* está aumentando. Além disso, a dificuldade do *SSRF* aumenta devido à complexidade dos serviços e arquiteturas em nuvem.

Figura 5 – Modelo de ataque *SSRF*



Fonte: (BANACH, 2021)

A Figura 5 é um exemplo de ataque tipo *SSRF*, ataques de *SSRF* ocorrem sempre que um aplicativo da *web* solicita um recurso remoto sem validar a *URL* fornecida pelo usuário. Um invasor pode forçar um aplicativo a enviar solicitações criadas para destinos inesperados, mesmo com proteção de *firewalls*, *VPNs* ou outros tipos de listas de controle de acesso à rede.

3 REVISÃO DA LITERATURA

Toda informação é influenciada por três propriedades principais: Confidencialidade, Integridade e Disponibilidade, além dos aspectos Autenticidade, Legalidade que complementam esta influência (SEMOLA, 2014).

A confidencialidade, integridade, disponibilidade, autenticidade e não repúdio de rede são essenciais para o bom desempenho das organizações no quesito de proteção, no qual o acesso e disponibilidade com que as informações cheguem para os usuários de forma íntegra e confiável. Para o autor, a Segurança da Informação é uma área que necessita de uma atenção especial e conhecimento aprofundados por parte dos profissionais responsáveis por manter a confidencialidade, integridade e disponibilidades das informações, garantindo que dados não sejam alterados, danificados e até mesmo extraviados (NAKAMURA; GEUS, 2007).

O caminho para se ter um ambiente considerado seguro não é fácil, mas com conhecimento adequado e realizando cada etapa de cada vez é propício para que o ambiente fique mais seguro (RIBEIRO, 2016).

Nakamura e Geus são referências pertinentes pois trouxeram vários conceitos relevantes para a fundamentação deste trabalho. Segundo o autor:

- Os vírus, *worms* e cavalo de Tróia, são ameaças constantes para as organizações proporcionando problemas sérios por serem incluídos em ataques distribuídos. Os *worms* diferencia dos vírus por espalhar sem a necessidade de interação do usuário, já cavalo de Tróia, são programas que aparentemente executam tarefas úteis mais na verdade utilizam de métodos prejudiciais para o sistema.

- vírus e *worms* utilizam meios para explorar vulnerabilidades de serviços ou de aplicativos, características essas que fazem como que sua disseminação seja vasta.

- Falta de estratégia de classificação das informações quanto ao valor e confiabilidade, resulta em um fator de risco para a organização, políticas de controle de acesso mal definidos, resulta com que usuários autenticados no início da conexão tenham acesso a diversas partes da rede interna, Informações que passam pela rede estão sujeitas a serem capturadas (NAKAMURA; GEUS, 2007).

(BASTA; BROWN, 2014) são importantes referências para este trabalho e trouxe conceitos relevantes. São eles:

- O *backdoors*, o *hacker* pode acessar uma rede, contornando o controle de *gateway* central depois de um ataque o invasor criar rota alternativa para raquear a rede novamente, ele pode por exemplo criar novo usuário com privilégios de administrador, um *malware* bem conhecido e o *rootkit* que cria *backdoor* na rede.

- Como o *backdoor* o cavalo de troia age de forma bem parecida, uma característica e entre os dois métodos de ataque e que eles trabalham de forma oculta, só que o cavalo

de troia necessita que o usuário instale de forma manual o vírus no computador, ele vem camuflado em anexos de email, *scripts* em *sites* fraudulentos dentre outros meios, todos esses meios de invasão do cavalo de troia usam a mente do próprio usuário para obriga-lo a fazer o que o agressor quer.

- As organizações precisam encontrar equilíbrio entre risco e benefício risco de um ataque *DoS* (*Denial of Service*) e o impacto, o custo de atualização do *hardware* pode ser maior que a perda potencial causada por um ataque *DoS*, o impacto potencial de um ataque desse poderia compensar o dinheiro gasto para atualizar sua tecnologia.

- O vazamento de informações é um problema que se torna ainda maior com o uso de redes sociais e outros serviços básicos na *web* que revelam informações úteis a *hackers* maliciosos, e que permitem descobrir os pontos fracos da organização é identifica as melhores técnicas e ferramentas possíveis para efetuar o ataque.

- Não é ilegal identificar uma empresa na Internet. É totalmente legal pesquisar todas as informações disponíveis sobre a empresa (incluindo número de telefone, horário comercial e endereço), e pode ser legal se o *hackers* não as *hackear*. Quando se procura informações dentro de uma empresa para fins de invasão é ilegal, quando o invasor instala *keylogger* ferramenta essa que captura as batidas no teclado do usuário é ilegal. Em uma máquina vulnerável é ilegal deixar um farejador, que intercepta e lê pacotes de dados em uma rede isso é ilegal.

- A engenharia social é um ato de fraude que tem o objetivo de enganar pessoas de boa-fé para que informem dados de sistemas não autorizados. O sucesso ou fracasso da engenharia social dependem do nível de persuasão por parte do *hacker* para manipular a psicologia humana (BASTA; BROWN, 2014).

De acordo com o *OWASP* o primeiro ataque mais comum no top 10 de 2021 é a quebra de Controle de Acesso, o qual garante que usuários não ultrapasse aquilo que foi permitido. Falhas no controle de acesso em aplicações podem deixar dados expostos levando a divulgação não autorizada de dados de usuários modificação ou destruição de todos os dados uma vez que o *hacker* pode utilizar de usuários administradores ou seja possuírem funções privilegiada (*OWASP*, 2021).

Jacques cita que o controle de acesso não elimina por completo o risco a segurança da informação, mas diminui em muitos casos a ocorrência de incidentes que comprometam a continuidade das atividades do negócio. Devido a preocupação com a segurança das informações tem-se a importância de se estabelecer na organização uma política de controle de acessos, que deve estar de acordo com os requisitos de negócio e de segurança, em uma boa política de segurança, todo acesso precisa ser considerado proibido, a menos que seja expressamente permitido. Deve-se também determinar as regras de acesso a diferentes sistemas, ambientes e serviços, bem como as responsabilidades dos usuários (MACHADO, 2012).

De acordo com o *OWASP* o segundo ataque mais comum no top 10 de 2021 é o

ataque de falha de criptografia, ao invés de atacar diretamente a criptografia, os atacantes tendem a roubar chaves, executar ataques *MitM* (*Man-in-the-Middle*) ou seja, o invasor intercepta mensagem entre as duas partes e depois se passa por uma das partes envolvida na troca de mensagem ou roubar dados em claro do servidor, em trânsito ou no cliente. Nos últimos anos esses ataques têm sido de maior impacto pois, a falta de criptografia ou criptografia fraca facilita para que o invasor faça o ataque. Segundo o *OWASP* para dados em trânsito, as vulnerabilidades no servidor são geralmente fáceis de detectar, enquanto os dados estáticos são difíceis. Quando se tem dados extremamente sigilosos os autores do *OWASP* indicam um sistema de criptografias mais robusto e consistente (OWASP, 2021).

De acordo com o *OWASP* o terceiro ataque mais comum no top 10 de 2021 é o ataque de injeção. São ataques bem comuns e com maior percentual de sucesso na internet, e o “pesadelo dos programadores” devido a seus numerosos tipos, grande superfície de ataque e a complexidade às vezes necessária para protegê-los (OWASP, 2021).

Os autores do *OWASP* usam a seguinte definição para ataques de injeção: Quando dados não confiáveis são enviados ao intérprete como parte de um comando ou consulta, podem ocorrer falhas de injeção, como injeção de *SQL*, *SO* e *LDAP*. Dados maliciosos de um invasor podem fazer com que o intérprete execute comandos inesperados ou acesse dados não autorizados (OWASP, 2021).

Segundo o autor, *SQL injection* de longe é a forma mais comum de ataque e extremamente mortal, ele opera injetando dados em uma aplicação web que são usadas em consultas *SQL*, quando bem sucedida pode-se manipular a consulta *SQL* para executar uma operação de banco de dados sem ser pretendida pelo programador (LOPES, 2018).

O *Cross-Site Scripting* (*XSS*) no *OWASP* 2021 também é considerado um ataque de injeção, um invasor pode usar *XSS* para enviar um *script* malicioso a um usuário, o navegador do usuário não tem como saber se o *script* não é confiável e executará o *script*. Por acreditar que o *script* veio de uma fonte confiável, o *script* malicioso pode acessar qualquer *cookie*, *tokens* de sessão ou outras informações confidenciais retidas pelo navegador (OWASP, 2021).

Segundo o autor, a consequência de um ataque *XSS* é a mesma, independentemente de ser armazenado ou refletido. A diferença está em como a carga chega ao servidor. O *XSS* pode causar vários problemas ao usuário final, variando de gravidade, de aborrecimento a danos gerais à conta. O ataque *XSS* mais sério envolve o vazamento do *cookie* de sessão do usuário, permitindo que o invasor sequestre a sessão do usuário e controle a conta (KIRSTENS, 2017).

De acordo com o (OWASP, 2021) edição de 2021, o quarto ataque catalogado é o ataque de *designer* inseguro que por sua vez é uma metodologia que avalia ameaças e garante que o código seja projetado e testado de forma robusta, para evitar ataques conhecidos.

Como o *OWASP* descreve, um design seguro pode ter erros de implementação

que levam a vulnerabilidades exploráveis. Em contraste, um design inseguro não pode ser melhorado mesmo com uma implementação perfeita porque a arquitetura básica do *design* não criou naturalmente os controles de segurança necessários para proteger contra certos ataques.

De acordo com o site (CECYBER, 2021), *design* inseguro é uma área que inclui a falta de proteção dos dados armazenados, problemas com programação lógica e exibição de conteúdo que revela informações confidenciais. Um dos maiores desafios para os profissionais de desenvolvimento é desenvolver um código que leve em consideração os objetivos de negócios e as necessidades de segurança para atingir esses objetivos.

De acordo com o OWASP o quinto ataque mais comum no top 10 de 2021 é o ataque de configurações incorretas de segurança, que podem ocorrer em aplicações *web*, servidores *web*, banco de dados. Quando atualizações não são instaladas, quando os *softwares* não são devidamente configurados, tem-se então, a ocorrência dessa vulnerabilidade. Isso geralmente é causado por configurações padrão inseguras, configurações incompletas ou *ad hoc*, armazenamento em nuvem aberto, cabeçalhos *HTTP (Hypertext Transfer Protocol)* configurados incorretamente e mensagens de erro detalhadas contendo informações confidenciais. Não apenas todos os sistemas operacionais, estruturas, bibliotecas e aplicativos precisam ser configurados com segurança, mas também precisam ser corrigidos e atualizados em tempo hábil (OWASP, 2021).

A partir da edição de 2021 do OWASP, o ataque de Entidades Externas de XML (XXE), passou a fazer parte do ataque de configurações incorretas de segurança. O ataque ocorre quando a entrada XML contendo uma referência a uma entidade externa é processada por um analisador XML com configuração fraca. Ele pode levar à divulgação de dados confidenciais que podem conter dados confidenciais como senhas ou dados privados do usuário, negação de serviço, falsificação de requisição junto ao servidor, e outros impactos no sistema (OWASP, 2021).

De acordo com o relatório da OWASP o sexto ataque mais comum no top 10 de 2021 é o ataque usando Componentes Vulneráveis. Alguns componentes podem não ter vulnerabilidades conhecidas quando criados, e pode haver resistência a atualizações por medo da quebra de funcionalidade ou perda de um recurso legado valioso, preocupações essas que são válidas, busca por vulnerabilidades de segurança pode resultar em perda de confiança por parte do cliente (OWASP, 2021).

De acordo com o OWASP o sétimo ataque mais comum no top 10 de 2021 é o ataque de falhas de identificação e autenticação, esta vulnerabilidade está relacionada com a autenticação e o gerenciamento de sessão da aplicação *web*, essa vulnerabilidade se caracteriza quando a aplicação *web* apresenta falhas de *logout*, expiração de sessão, sistemas de “lembre-me” (OWASP, 2021).

De acordo com o (OWASP, 2021) o oitavo ataque mais comum no top 10 de 2021 é o ataque falhas de integridade de software e dados estão relacionadas a código

e infraestrutura que não protegem contra violações de integridade. Um exemplo disso é quando um aplicativo depende de *plugins*, bibliotecas ou módulos de fontes não confiáveis, repositórios e redes de entrega de conteúdo, muitos aplicativos agora incluem atualização automática, em que as atualizações são baixadas sem verificações de integridade suficientes e aplicadas a um aplicativo anteriormente confiável. Os invasores podem baixar suas próprias atualizações para distribuir e executar em todas as instalações.

Outro exemplo é quando objetos ou dados são criptografados ou serializados em uma estrutura que pode ser vista e modificada por um invasor e são vulneráveis à serialização desprotegida. Desserialização Insegura, que é o processo de transformar um objeto em um formato de dados que pode ser restaurado posteriormente. Os dados são armazenados em uma determinada estrutura usando formatos como *JSON* ou *XML*. Muitas linguagens de programação fornecem serialização nativa ou permitem processos de serialização personalizados, que podem então ser usados de forma maliciosa. Desserialização insegura tem sido encontrada para permitir o código remoto de execução, negação de serviço, repetição, injeção e escalada de privilégios (OWASP, 2021).

Segundo o relatório da OWASP o nono ataque mais comum no top 10 de 2021 é o ataque usando As falhas de registro e monitoramento de segurança. Embora a maioria das aplicações seja criada para registrar algum nível de controle e informação de acesso, em aplicação de rede muitas vezes não se faz um monitoramento de acesso, sendo assim logs de acesso não são coletados e nem analisados, com isso a dificuldade de detectar uma violação, e se recuperar de um caso de incidente. Os invasores usam pontos cegos para não serem detectados e penetrar em aplicativos e redes. O monitoramento cuidadoso desses dados pode ajudá-lo a detectar ataques precocemente, permitindo que se estabeleça ou até mesmo aprimore uma postura defensiva e, em última análise, minimize o dano ou impacto para as organizações. (OWASP, 2021).

De acordo com o (OWASP, 2021) o décimo tipo de ataque elencado no ranking de 2021 é falsificação de solicitação do lado do servidor (*SSRF*). De acordo com (PORTSWIGGER, 2022) A falsificação de solicitação do lado do servidor (também conhecida como *SSRF*) é uma vulnerabilidade de segurança da *web* que permite que um invasor induza o aplicativo do lado do servidor a fazer solicitações para um local indesejado. Em um ataque *SSRF* típico, o invasor pode fazer com que o servidor faça uma conexão com serviços somente internos dentro da infraestrutura da organização. Em outros casos, eles podem forçar o servidor a se conectar a sistemas externos arbitrários, vazando dados confidenciais.

Segundo (PORTSWIGGER, 2022) ataques *SSRF* bem-sucedidos geralmente podem levar a ações não autorizadas e acesso a dados dentro da organização, seja no próprio aplicativo vulnerável ou em outros sistemas de *back-end* com os quais o aplicativo pode se comunicar. Em algumas circunstâncias, uma vulnerabilidade no *SSRF* pode permitir que um invasor execute comandos arbitrários.

De acordo com (BANACH, 2021) os invasores podem explorar vulnerabilidades

no *SSRF* para lançar ataques por meio de servidores *web* vulneráveis da seguinte maneira:

- O invasor envia uma solicitação falsa a um servidor da *web* vulnerável ao *SSRF* e que é hospedado na mesma rede interna do servidor de destino.
- O servidor *web* vulnerável envia a solicitação controlada pelo invasor ao servidor da vítima, ignorando o *firewall*.
- O servidor da vítima responde ao servidor *web* com os dados solicitados.
- Se a vulnerabilidade específica do *SSRF* permitir, os dados serão enviados de volta ao invasor.

4 METODOLOGIA

Esta seção tem como objetivo expor a metodologia utilizada para construção desse trabalho.

Para o levantamento teórico utilizamos sites e livros com afincos, para desenvolver os testes com mais precisão e obter êxito nos testes realizados.

O principal critério para a escolha das ferramentas usadas nos laboratórios de testes foram sua disponibilidade e aplicação, para desenvolver testes em aplicações web, desenvolvendo ataques que são mais recorrentes de acordo com o *OWASP* e *CVEdetails*. Todas as versões das ferramentas utilizadas são *opensource* com atualizações mais recentes e de fácil acesso. As ferramentas utilizadas foram *Burp Suite*, *OWASPZAP*, ferramentas essas podendo ser instaladas nos sistemas operacionais *Windows* ou *Linux*. O *Burp Suite* e o *OWASPZap* são ferramentas distintas, as características do *Burp* é que ele faz a interceptação de pacotes entre o usuário e o servidor permitindo realizar testes de penetração e segurança da aplicação. Logo o *OWASPZap* realiza somente varreduras na aplicação em busca de vulnerabilidades.

Para cada vulnerabilidade foram adotados laboratórios de maneira mais prática para reproduzi-las, para algumas foram necessárias a criação de ambientes *localhost*. Para os resultados foram utilizados os laboratórios *WebSecurity Academy*, *Vulnweb*, *Portswigger* e *Acunetix*. Organizações que tem por foco o ensino de segurança de aplicações web, algumas vulnerabilidades por serem tanto quanto complexas, os ambientes citados acima serviram de apoio para os testes realizados.

Os sites utilizados nos testes de vulnerabilidade são propositalmente vulneráveis a ataques, para que possamos estudá-los e resolver as vulnerabilidades.

Para as vulnerabilidades localizadas pelas ferramentas explicamos teoricamente o processo por trás de cada ataque, como ele falhou e o que sua exploração bem-sucedida pode significar para sua aplicação. Como fonte de informação, os conteúdos da *Web Security Academy* e *Acunetix* foram utilizados.

Baseado na Metodologia de Classificação de Risco da *OWASP*, a Figura 6 ressalta probabilidade de ocorrência e impacto técnico em uma aplicação usando o seguinte esquema de classificação.

Figura 6 – Classificação de risco

Agente Ameaça	Abuso	Prevalência da Falha	Detetabilidade	Impacto Técnico	Impacto Negócio
Específico da Aplicação	Fácil: 3	Predominante: 3	Fácil: 3	Grave: 3	Específico do Negócio
	Moderado: 2	Comum: 2	Moderado: 2	Moderado: 2	
	Difícil: 1	Incomum: 1	Difícil: 1	Reduzido: 1	

Fonte: Owasp (2021)

Cada organização é única, assim como os agentes de ameaça. Se organizações de interesse público usam aplicações *CMS* para obter informações públicas e sistemas de saúde usam o mesmo *CMS* para obter registros de saúde confidenciais, o agente de ameaça e o impacto comercial desse *software* serão muito diferentes. É importante entender os riscos enfrentados pela organização não apenas com base no agente de uma ameaça específica, mas também com base no impacto nos negócios. Portanto, o trabalho do profissional (e da equipe) de segurança de informação, redes ou sistemas é no sentido de que o impacto dos ataques sejam os menores possíveis (OWASP, 2021).

5 IMPLEMENTAÇÃO

Nesta seção tem como objetivo demonstrar em detalhes ferramentas utilizadas para realizar testes de penetração e análise de vulnerabilidades em aplicações *web*.

5.1 OwaspZap 2.11.1

O *Zed Attack Proxy (ZAP)* é uma ferramenta gratuita de teste de penetração de código aberto do *Open Web Application Security Project (OWASP)*. O ZAP foi projetado especificamente para testar aplicativos da *Web* sendo flexível e extensível. Em sua essência, o ZAP é o que é conhecido como “*proxy man-in-the-middle*”. Como exemplificado na Figura 7 ele fica entre o navegador do testador e o aplicativo da *web* para que possa interceptar e inspecionar as mensagens enviadas entre o navegador e o aplicativo da *web*, modificar o conteúdo, se necessário, e encaminhar esses pacotes para o destino. Ele pode ser usado como um aplicativo independente (OWASPZAP, 2022).

Figura 7 – Modelo de teste OWASP ZAP

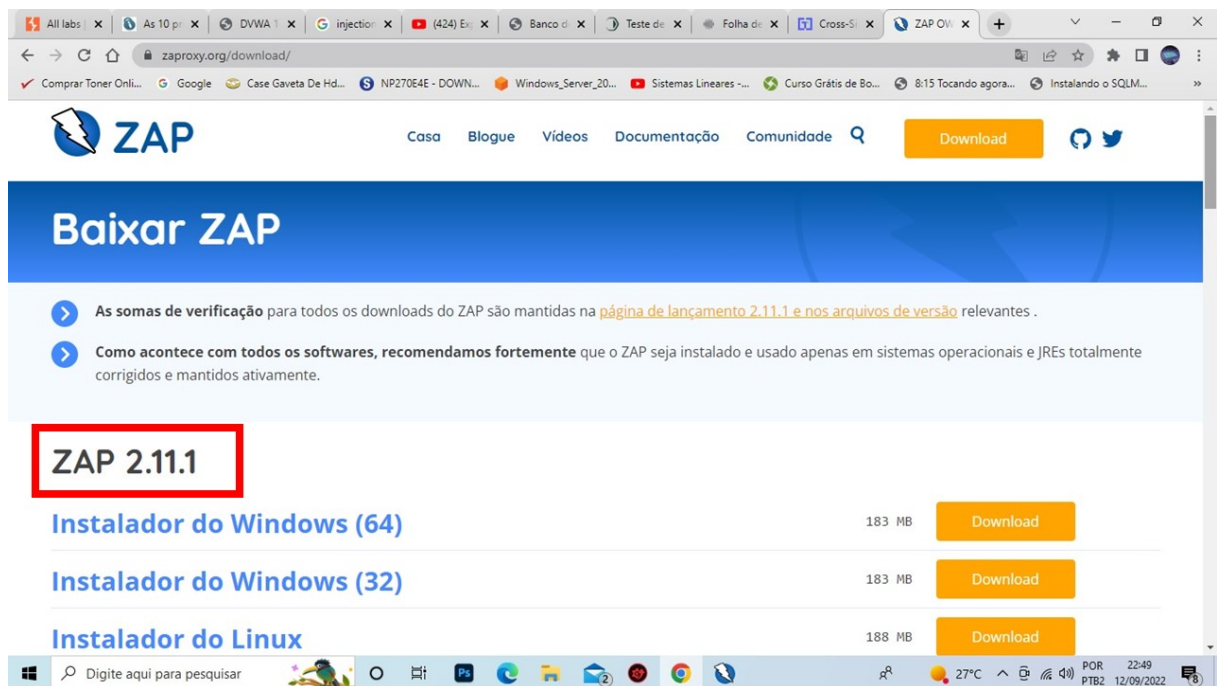


Fonte: OwaspZAP (2022)

O ZAP oferece recursos para diferentes níveis de habilidade, de desenvolvedores a novos testadores. O ZAP tem versões para *Windows*, *Linux* e *Mac*. Funcionalidades adicionais estão disponíveis gratuitamente em vários *add-ons* no ZAP Marketplace que podem ser acessados a partir do cliente ZAP. ZAP é de código aberto, então você pode inspecionar o código-fonte para ver exatamente como os recursos são implementados.

1º Passo: foi realizado o *download* do OwaspZap versão 2.11.1 no site <https://www.zaproxy.org/download/> representando na Figura 8.

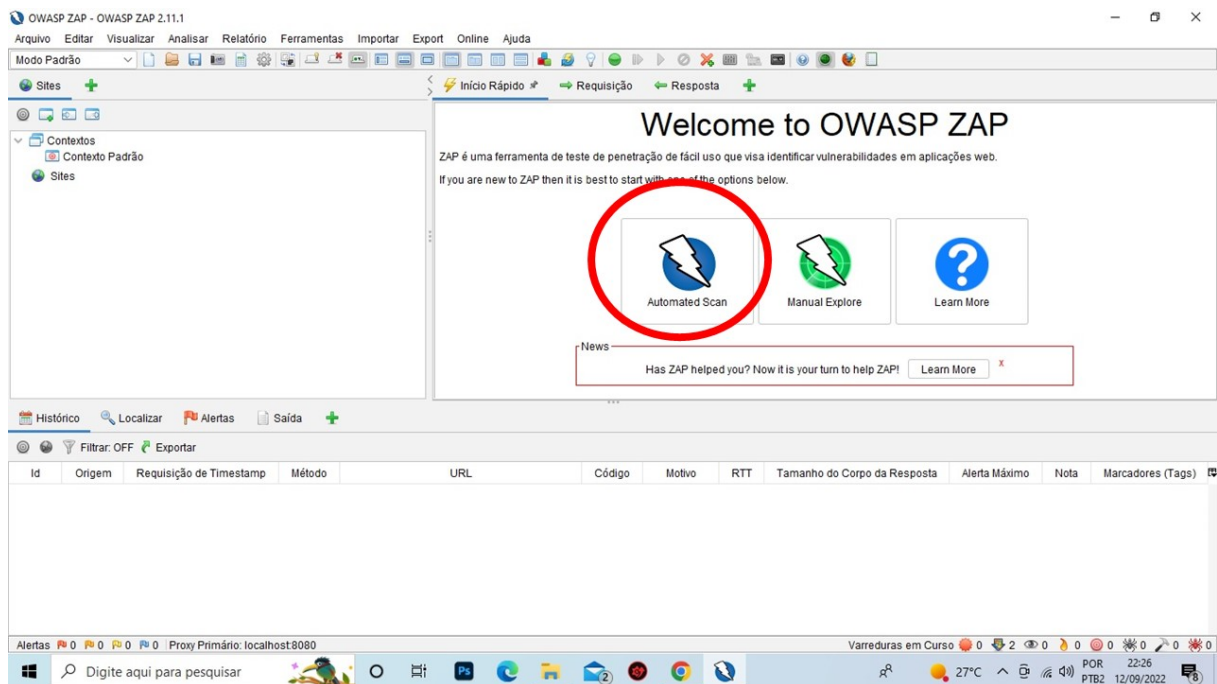
Figura 8 – Download do OWASPZap



Fonte: Próprio autor, 2022

2º Passo: Neste processo utilizamos o processo de *scanner* automatizado.

Figura 9 – Tela inicial do OWASPZap

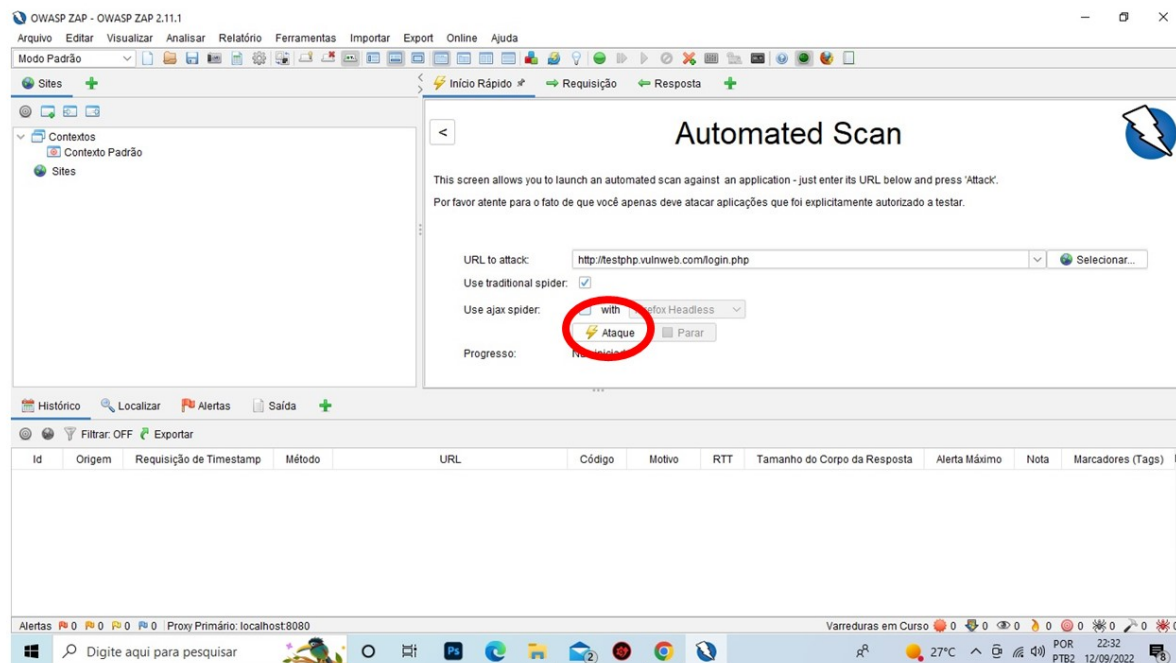


Fonte: Próprio autor, 2022

3º Passo: Como na Figura 10 na parte onde está *URL to attack* utilizamos o site

de teste do laboratório da *acunetix*¹ <http://testphp.vulnweb.com/login.php> e iniciamos o ataque clicando no botão Ataque.

Figura 10 – Scanner automático OWASPZap

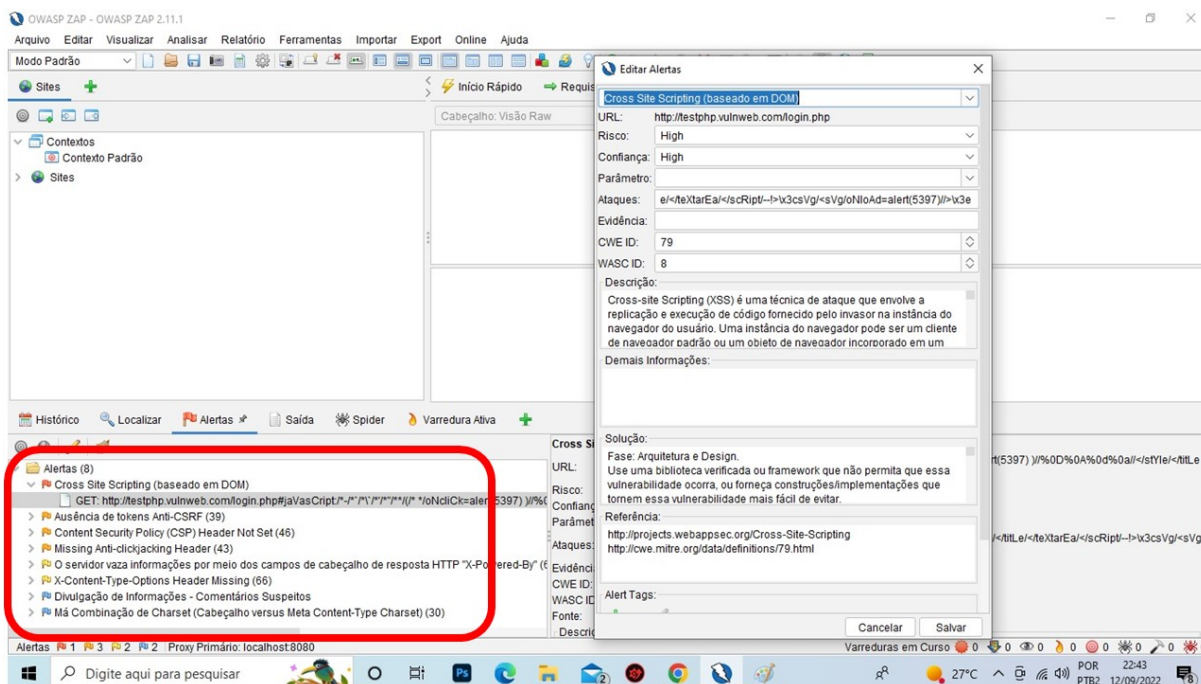


Fonte: Próprio autor, 2022

4º Passo: Após aproximadamente 45 min do início do ataque foram detectados 8 alertas de vulnerabilidades, enumeradas da mais crítica para menos crítica de acordo com a cor do ícone da bandeira de alerta, como na Figura 11 o ícone da bandeira de cor vermelha e a mais crítica e a bandeira de cor azul e a menos crítica. O primeiro alerta mais crítico é o ataque de *Cross Site Scripting (XSS)* que de acordo com (DETAILS, 2022) este tipo de ataque foram catalogados 23365 vezes desde o ano de 1999, é 2357 vezes somente no ano de 2022.

¹O *Acunetix* é o *scanner web* muito utilizado por *hackers*, administradores de sites, *webmasters* e analistas de segurança, com esse *scanner* consegue se fazer varreduras completas em um *sites* em busca de vulnerabilidades. A *acunetix* disponibiliza um exemplo de *PHP* que é intencionalmente vulnerável a ataques na web que pode-se utiliza-lo para testar ferramentas e habilidades manuais de *hacking*.

Figura 11 – Resultado do teste



Fonte: Próprio autor, 2022

5.2 Quebra de autenticações.

No teste abaixo utilizamos a ferramenta chamada *BurpSuite community edition* versão gratuita desenvolvida pela *portswigger*².

De acordo com (COOPER, 2022) O *Burp Suite* fornece ferramentas de teste de penetração em aplicativos da *web*. É possível utilizar a versão paga ou a versão gratuita do *Burp Suite*. A versão gratuita é chamada *Community Edition* que é utilizada para realizar testes manualmente de sites ou serviços da *web*. As duas versões pagas do *Burp Suite* são a *Professional Edition* e a *Enterprise Edition*. A *Enterprise Edition* é totalmente automatizada e atende à definição de um gerenciador de vulnerabilidades. Assim como a *Free Edition*, a *Professional Edition* destina-se ao uso em testes de penetração. No entanto, é mais automatizado do que o *Community Edition*.

Processo de instalação do *Burp Suite Community Edition*:

- Acessar a página <https://portswigger.net/burp/releases/professional-community-2022-8-4?requestededition=communityrequestedplatform=> e clique no botão *Download*. Isso leva você para outra página.
- Selecione seu sistema operacional e clique no botão *Download*.
- Clique no arquivo baixado para executar o instalador.

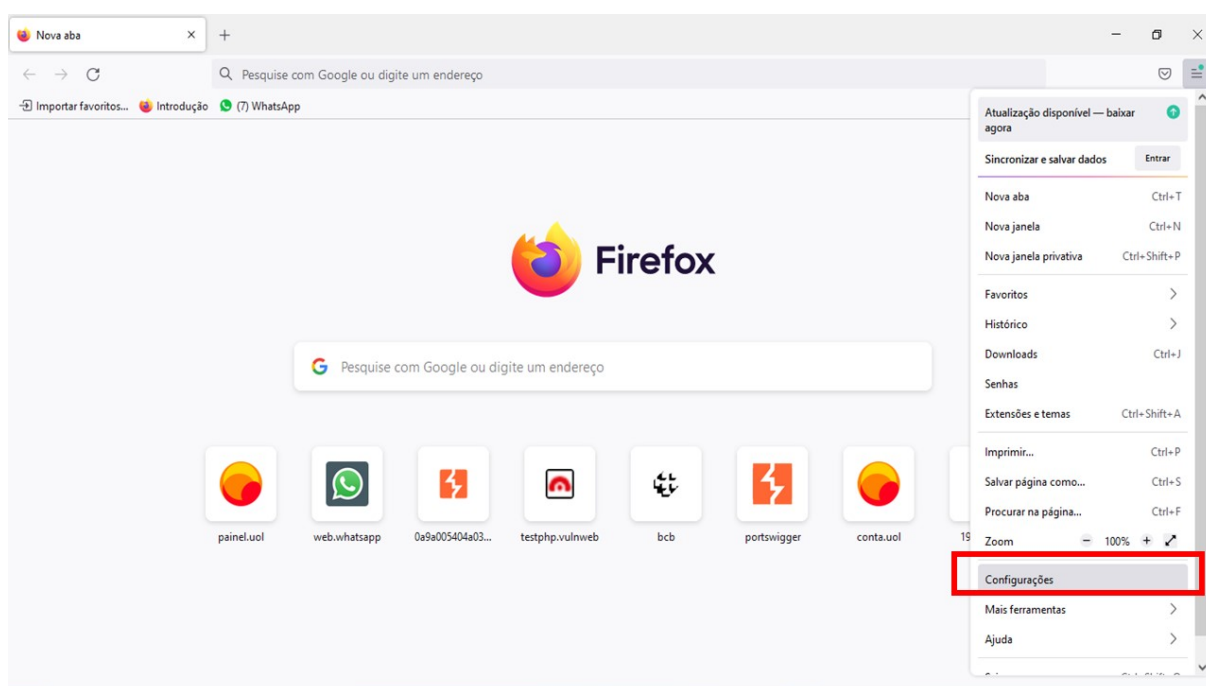
²O *PortSwigger Web Security* é um líder global na criação de ferramentas de *software* para testes de segurança de aplicativos da *web*. Por mais de uma década, trabalham na vanguarda do setor de segurança na *web*, e seu conjunto de ferramentas é bem estabelecido como o kit de ferramentas padrão usado pelos profissionais de segurança da *web*.

- Siga as instruções no assistente de *download*, percorrendo cada página pressionando o botão Avançar.

Nos exemplos abaixo utilizamos o *BurpSuite* versão 2022.8.2 para *Windows*, o navegador mozilla e o site de teste do laboratório da *acunetix* <http://testphp.vulnweb.com/login.php>. Utilizamos o site para fazer a quebra de senha utilizando o ataque de força bruta.

1º Passo: Primeiro passo configuramos o *proxy* no navegado Mozilla para que o *Burp* possa interceptar as requisições que o site faz para o servidor. Primeiramente abrimos o navegado Mozilla Firefox e clicamos no menu e selecionamos configurações como indicado na Figura 12.

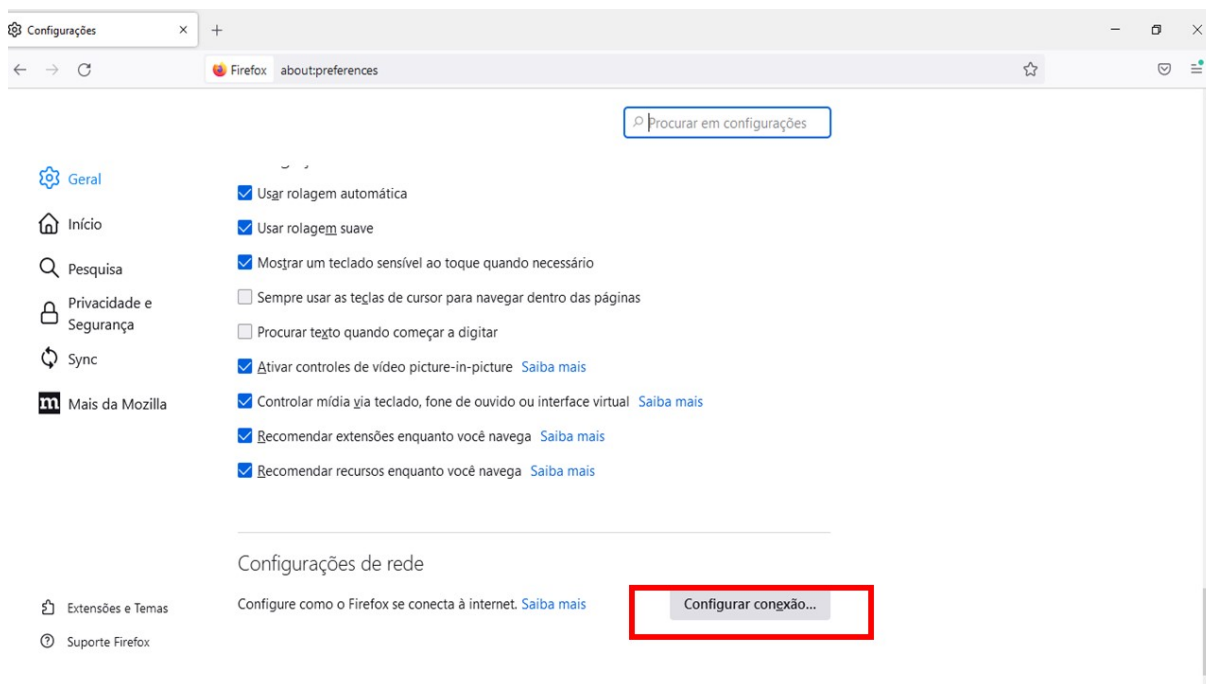
Figura 12 – Mozilla Firefox



Fonte: Próprio autor, 2022

2º Passo: No menu geral no item configurações de rede clica em configurar conexão.

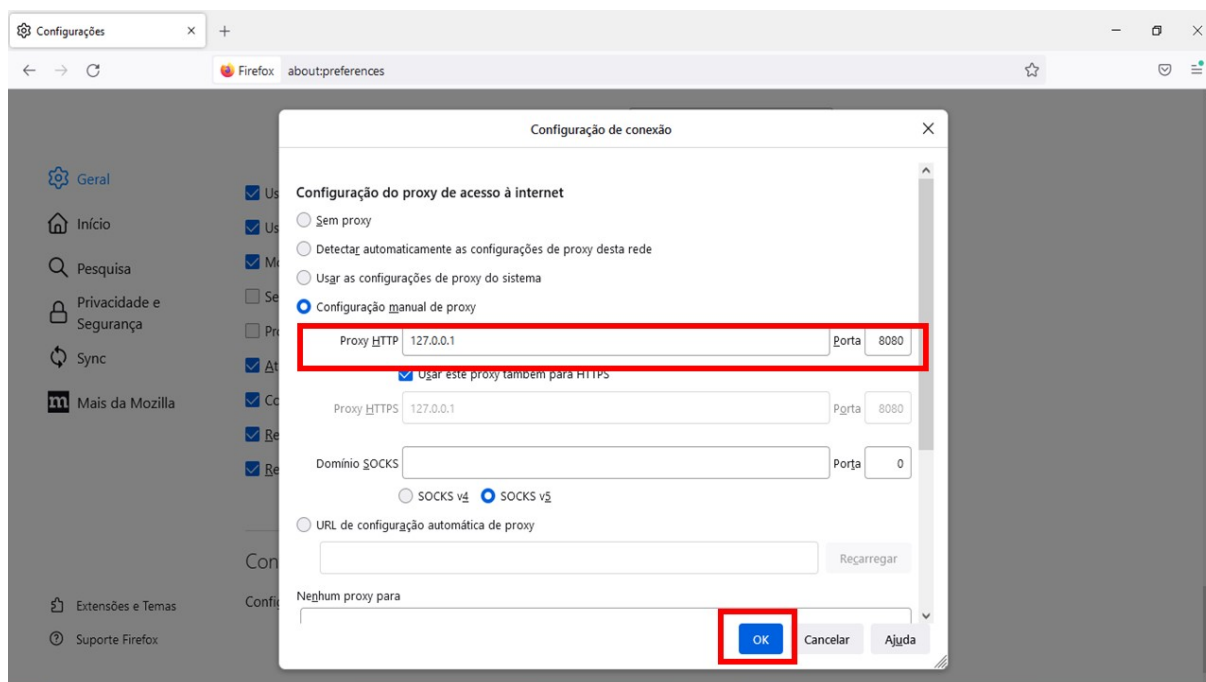
Figura 13 – Tela de configuração de conexão Mozilla Firefox



Fonte: Próprio autor, 2022

3º Passo: Na tela de configuração de conexão mais a opção configuração manual de *proxy*, no campo *Proxy HTTP* utilizamos o ip *localhost* 127.0.0.1 e no campo porta utilizamos a porta 8080.

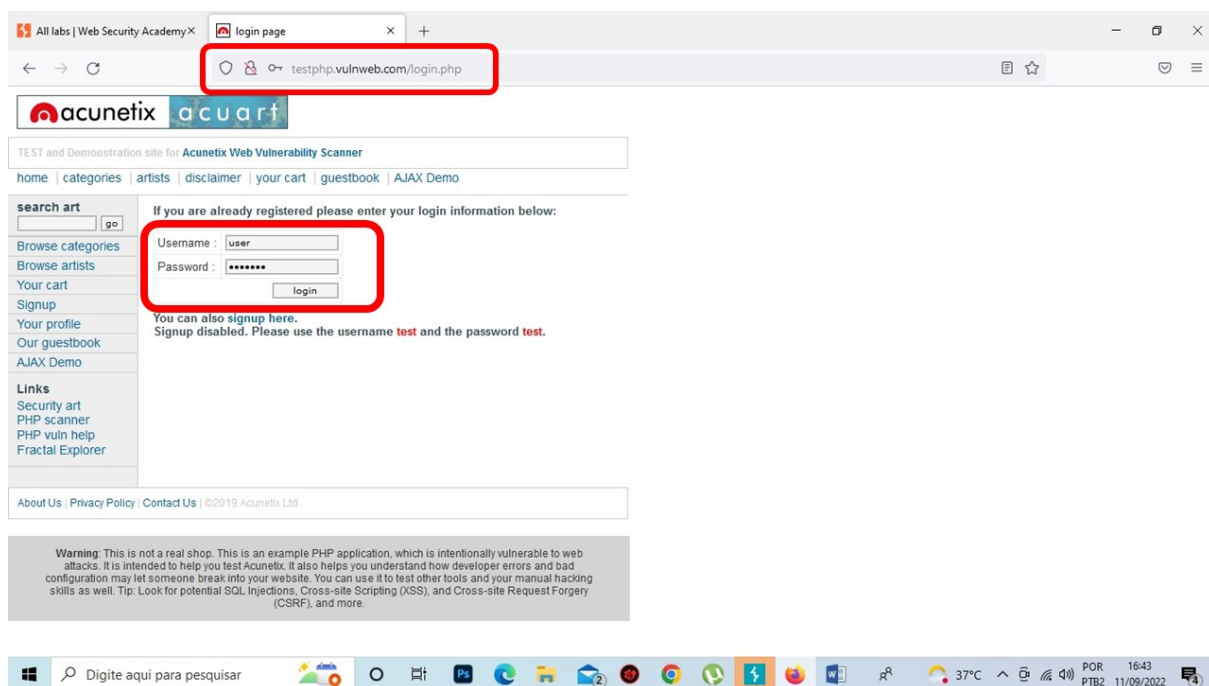
Figura 14 – Configuração de proxy Mozilla Firefox



Fonte: Próprio autor, 2022

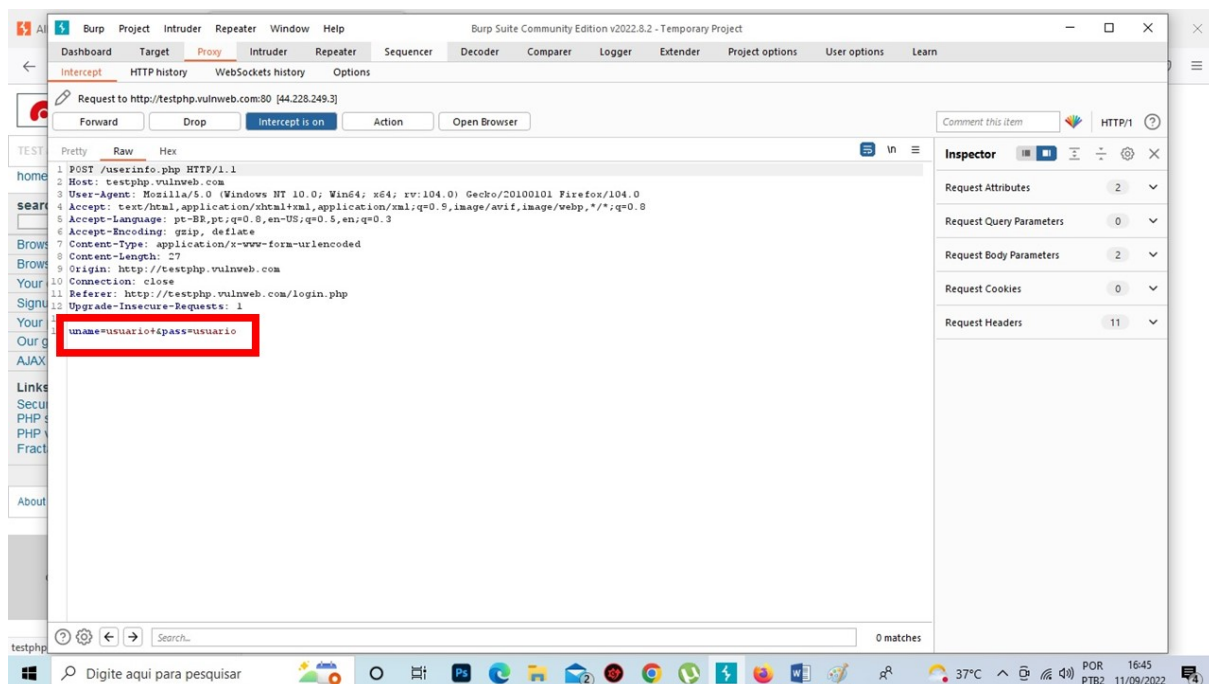
4º Passo: Após a configuração do *Proxy* do *BurpSuite* no navegador Mozilla Firefox, abrimos o *BurpSuite* e o navegador mozilla, entramos no site de teste do laboratório da *acunetix* <http://testphp.vulnweb.com/login.php>. Neste processo usaremos *username* e *password* aleatório para que possamos identificar os campos dentro do *burpsuite* quando fizermos a interceptação do pacote.

Figura 15 – Laboratório Acunetix



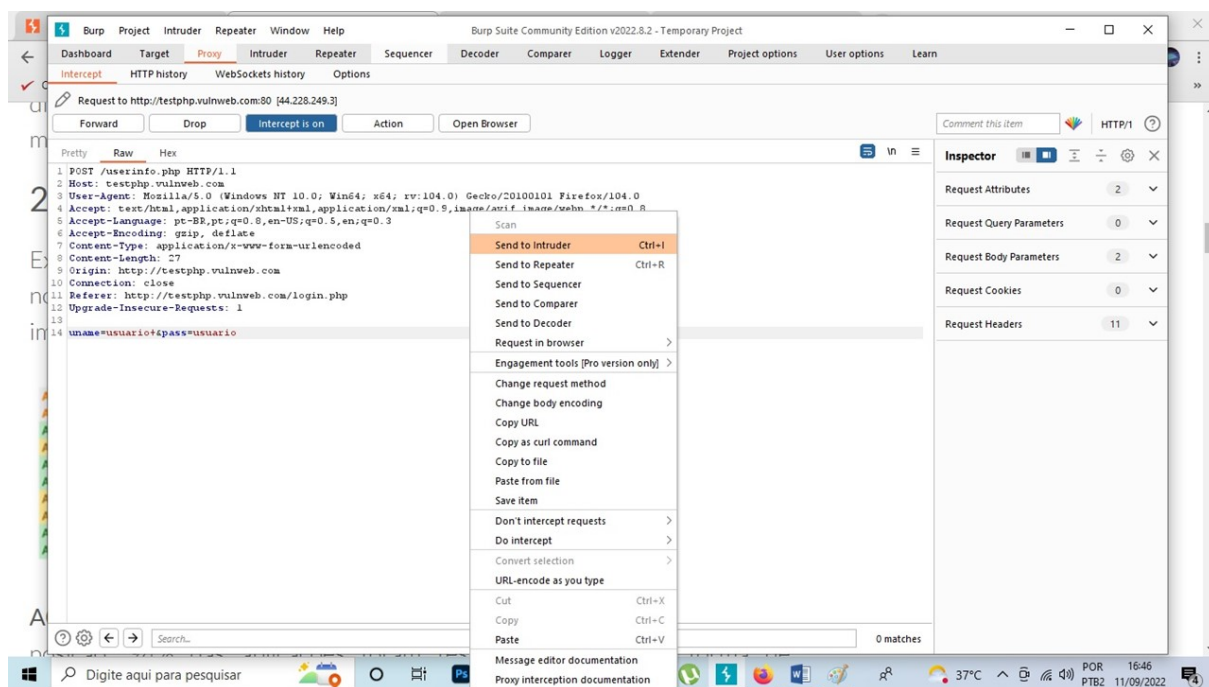
Fonte: Próprio autor, 2022

5º Passo: Dentro do *BurpSuite* através da interceptação localizamos os campos digitados nos campos *username* e *password* como apresentado na Figura 16.

Figura 16 – Pacote interceptado pelo *Burp Suite*

Fonte: Próprio autor, 2022

6º Passo: Após localizado os campos de usuário e senha interceptados pelo *BurpSuite*, clica com o botão direito do mouse em qualquer área, seleciona a opção *send to intruder*.

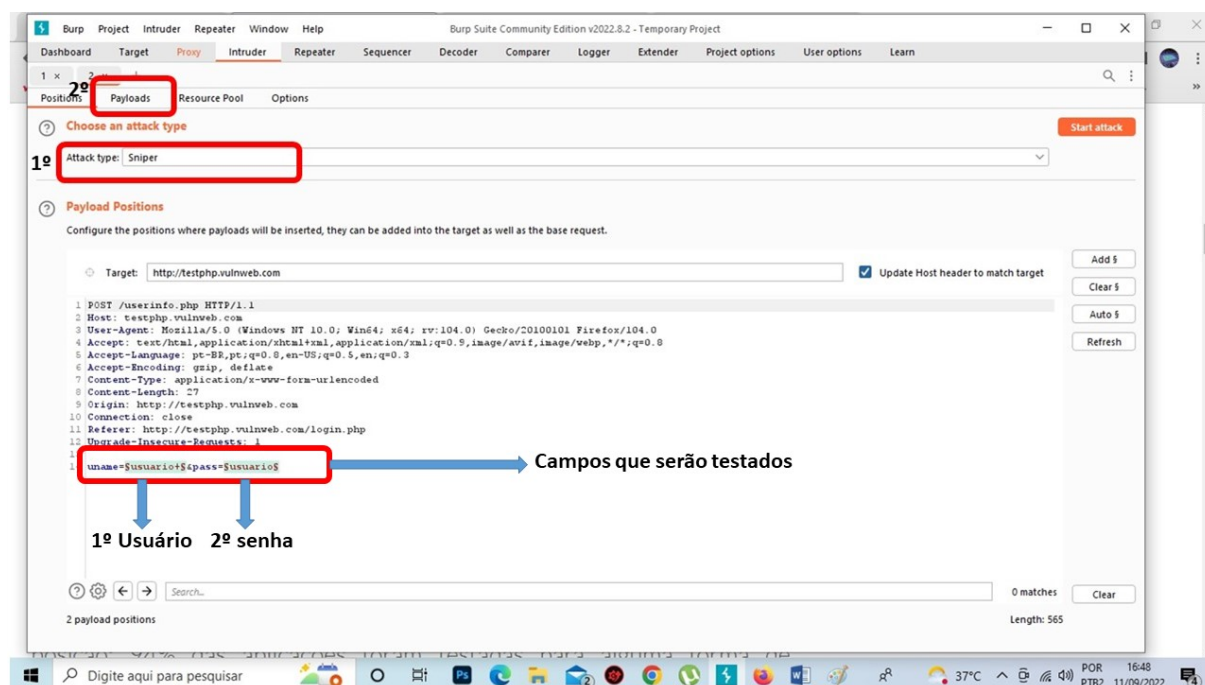
Figura 17 – Início do ataque pelo *Burp Suite*

Fonte: Próprio autor, 2022

7º Passo: Dentro da opção *Intruder*, selecionamos o tipo de ataque (*Sniper*). Segundo (MORENO, 2017) a opção de ataque *sniper* é similar a um *sniper*, cada “tiro” (ou *payload*) é feito um após o outro, recomendado para situações onde se queira testar campos de forma individual. De acordo com a imagem abaixo selecionamos a aba *Payloads*.

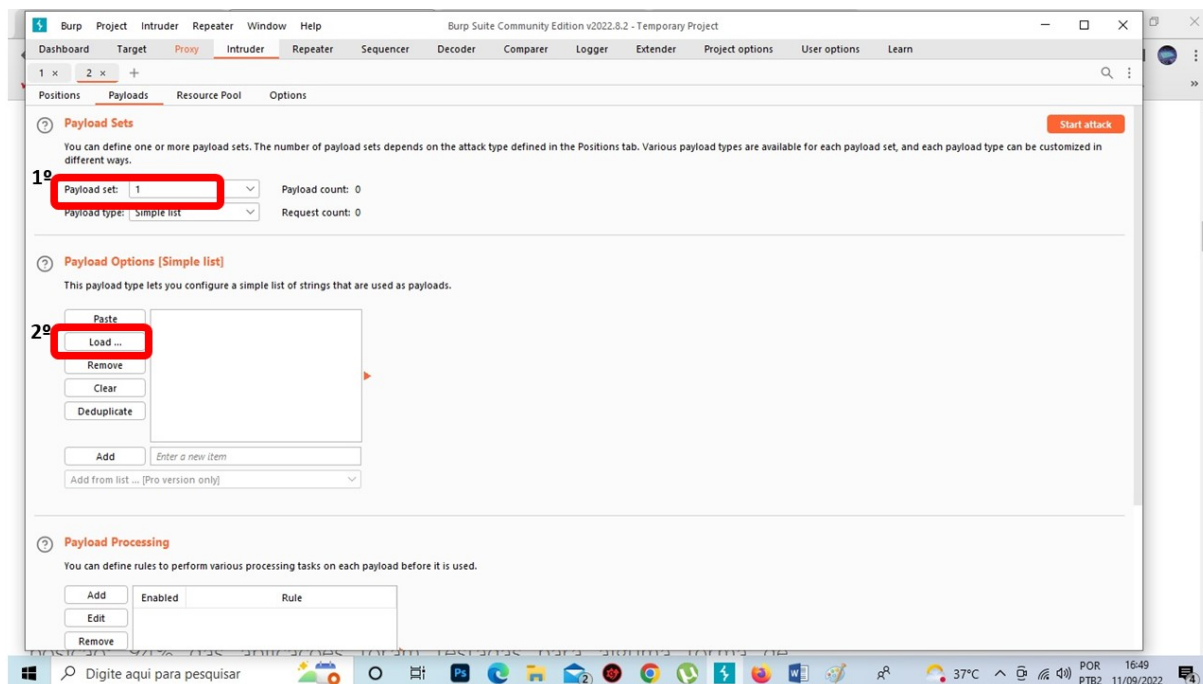
Segundo o site (HACKERSEC, 2018) o termo carga útil (*payload*) refere-se a carga de códigos malicioso que executa ações destrutivas em um sistema de destino e fornece acesso e permissões privilegiados. Por exemplo, criar usuários, iniciar ou migrar processos, excluir arquivos na fase de pós-exploração dentre outros. Um exemplo de (*payload*) é uma (*wordlist*) que contem vários usuários e senhas para se testar pelo método de injeção em uma aplicação.

Figura 18 – Localização dos campos que serão injetados *payloads*

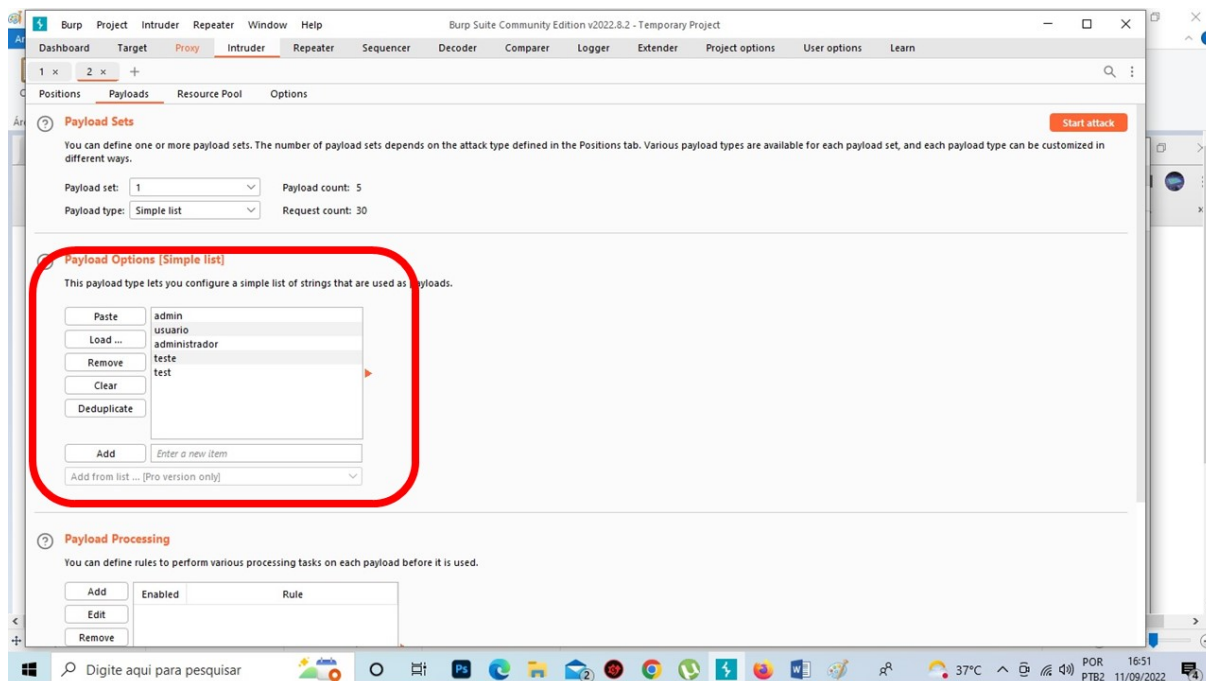


Fonte: Próprio autor, 2022

8º Passo: Neste processo foi escolhido o número de *payloads*, no caso testado serão 2 *payloads*, foram criadas 2 listas no bloco de notas do *Windows*, uma lista com exemplos de nomes de usuários e outra lista com exemplos de senhas predefinidas. Foi escolhido a 1º *payload*, clica em *load* para carregar a lista de *username* que testara o primeiro campo como pode ser observado nas Figuras 19 e 20.

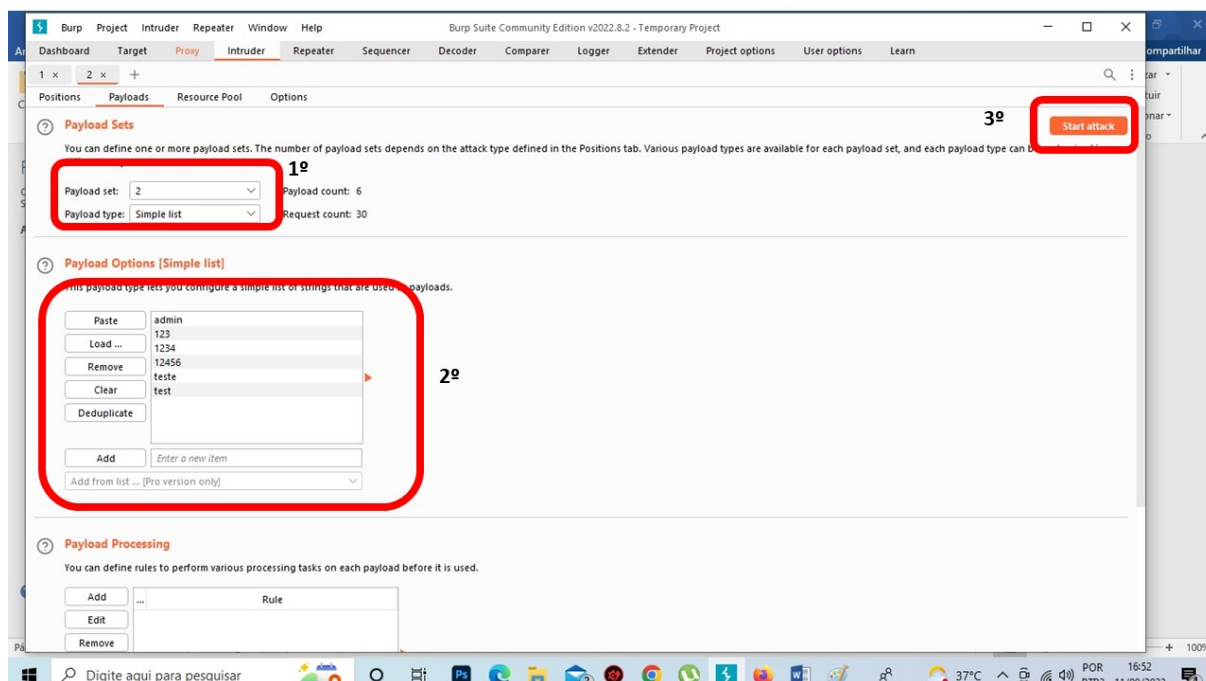
Figura 19 – Escolha do primeiro *payload*

Fonte: Próprio autor, 2022

Figura 20 – *Wordlist* de usuários

Fonte: Próprio autor, 2022

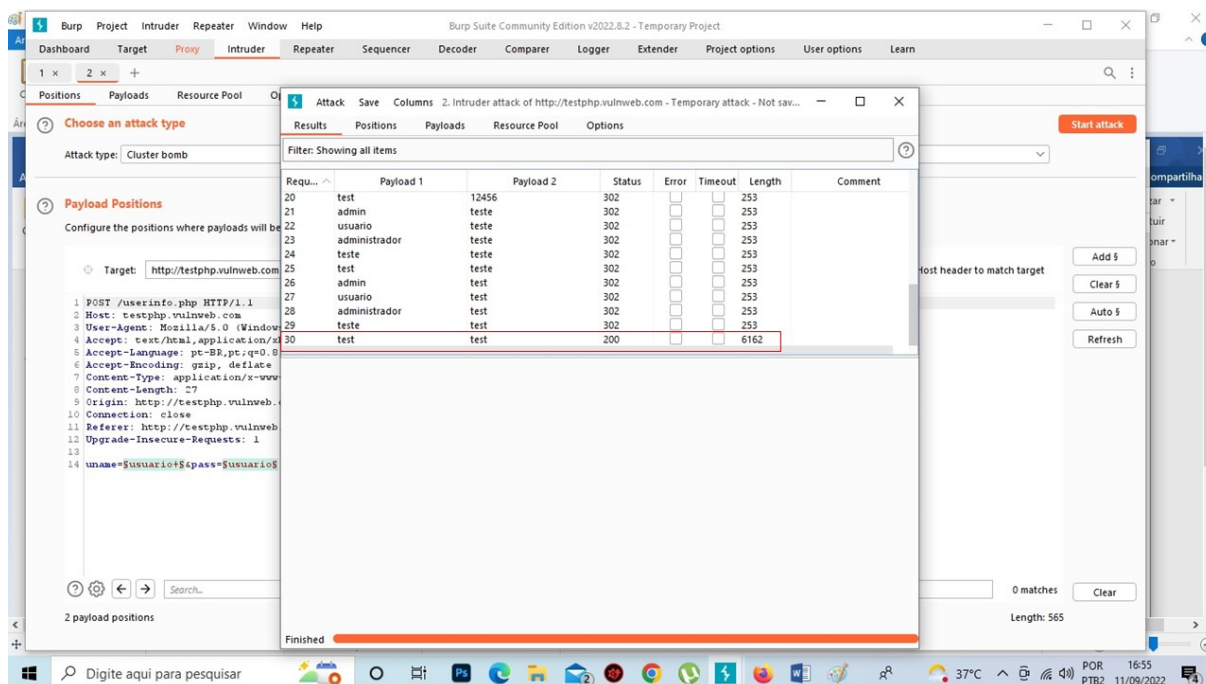
9º Passo: Neste processo foi escolhido o segundo *payload* que testara o segundo campo que corresponde a senhas, após carregar a lista de senhas clica se em *start attack* que iniciará imediatamente o ataque ao *site* de teste.

Figura 21 – Escolha do segundo *payload*

Fonte: Próprio autor, 2022

10º Passo: Após iniciado o ataque será exibida a tela abaixo, que no teste realizado foi encontrado o usuário e senha correspondente de *login* ao site, para se saber se o teste de usuário e senha foi bem sucedido, o *BurpSuite* enumera códigos, no exemplo abaixo na coluna *Status* o único código diferente dos outros e o de número 200 que significa que obteve resposta positiva quanto ao ataque, e a coluna *length* corresponde ao tamanho da página após o ataque bem sucedido. A seguir na Figura 22 o nome de usuário é *test* e a senha é *test*.

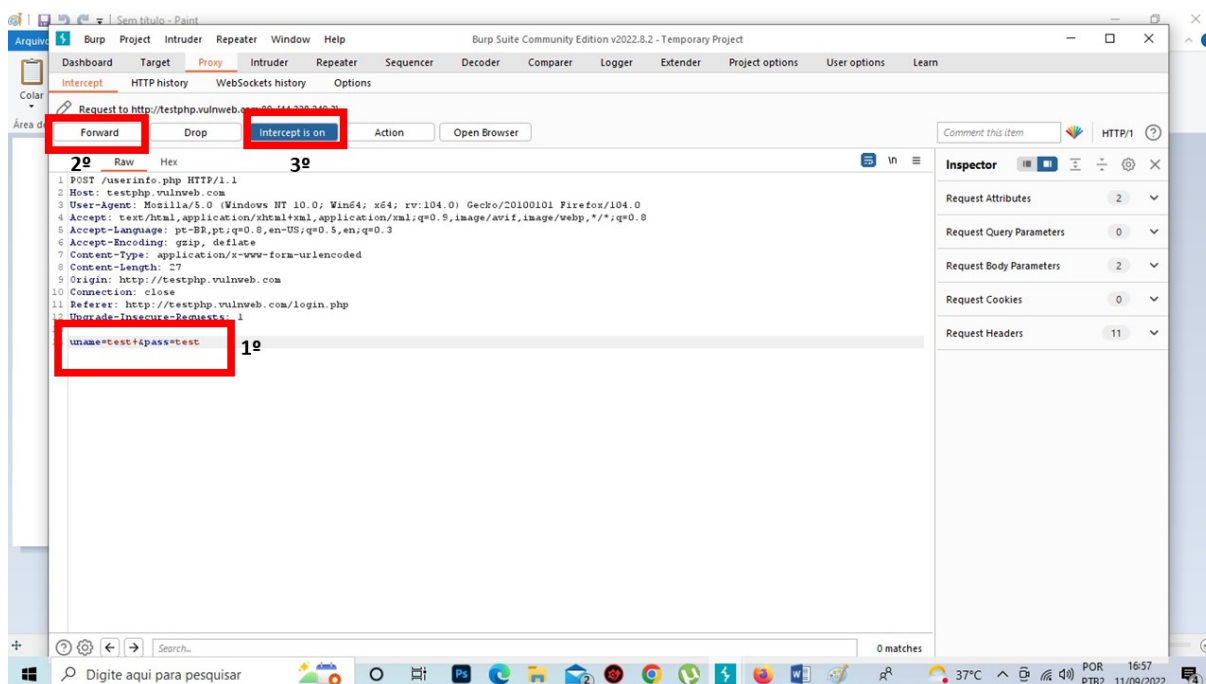
Figura 22 – Resultado do ataque



Fonte: Próprio autor, 2022

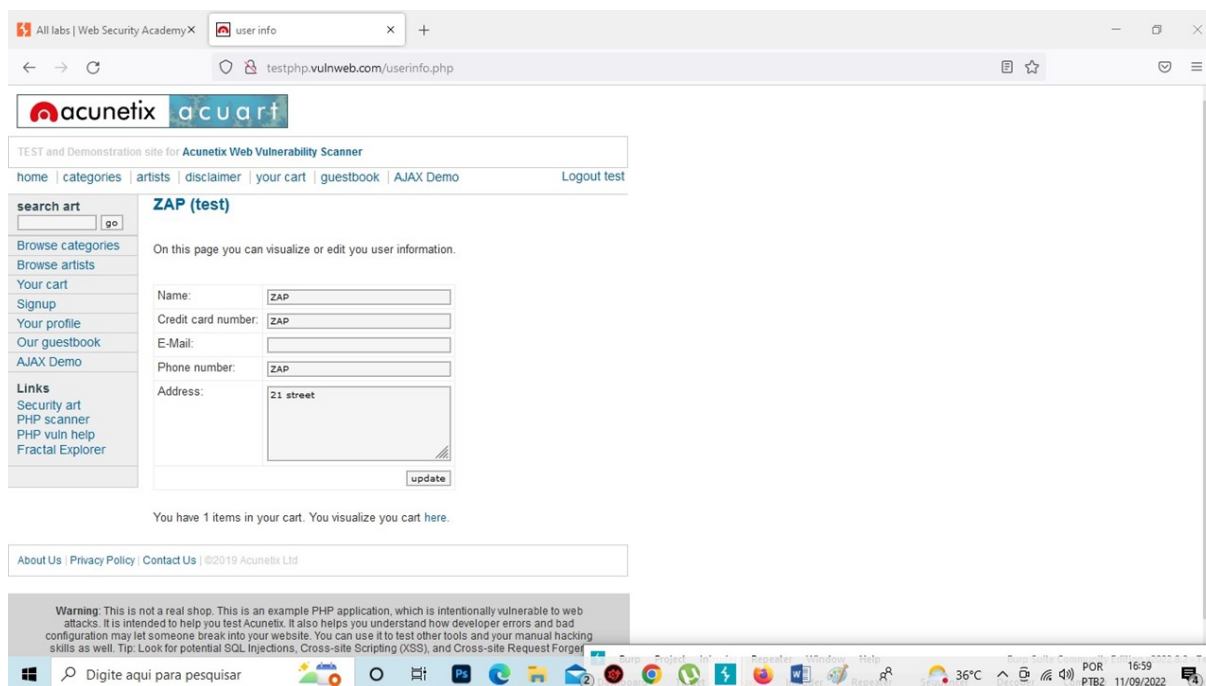
11º Passo: Neste passo retornamos ao menu *proxy* e no sub menu *intercept*, primeiramente alteramos o campo *uname* para o usuário encontrado no passo anterior e no campo *pass* para a senha encontrada no processo anterior. Após fixados o nome de usuário e senha clica se em *forward* que aceitara a requisição, e marcar a opção *intercept is on* pra alterar para *intercept is off*.

Figura 23 – Fixados campos de usuário e senha



Fonte: Próprio autor, 2022

12º Passo: Nesse último passo o acesso ao site de teste foi realizado com sucesso.

Figura 24 – Acesso ao laboratório *Acunetix* realizado com sucesso

Fonte: Próprio autor, 2022

5.3 Ataque de injeção utilizando BurpSuite

Segundo (MORENO, 2017) o ataque de injeção é um tipo de ataque que tenta injetar dados em consultas ou comandos do sistema exemplo de injeção: Injeção *SQL*, *LDAP*, *XML*, *HTML*, *etc.*

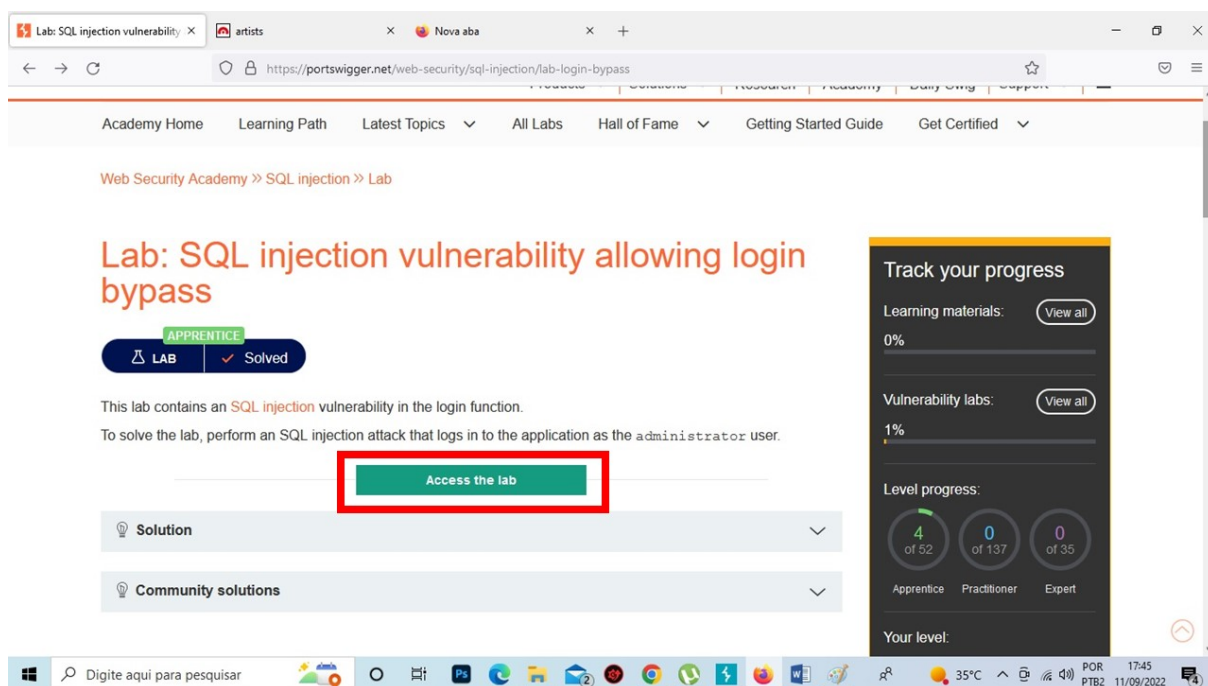
De acordo com (OWASP, 2021) o ataque de injeção ocupou a 3º posição no *ranking* como ilustrado na Figura 4, logo o (DETAILS, 2022) o ataque de injeção ocupou o 6º lugar como monstado nas figuras 1 e 2, sendo catalogados 1353 ataques no ano de 2022, e 10554 desde 1999.

Com o *BurpSuite* devidamente instalado e configurado de acordo com o exemplo anterior, iremos mostrar abaixo o método de ataque de injeção, como é feito, bem como o modo de prevenir esse tipo de ataque.

Nos exemplos abaixo é mostrado um exemplo de ataque de injeção que foi explorado a vulnerabilidade de injeção de *SQL* que permite o *bypass* de *login* realizado no laboratório de teste do site *portswigger*, juntamente com o *BurpSuite*.

1º Passo: No primeiro passo foi iniciado o laboratório no site <https://portswigger.net/web-security/sql-injection/lab-login-bypass>

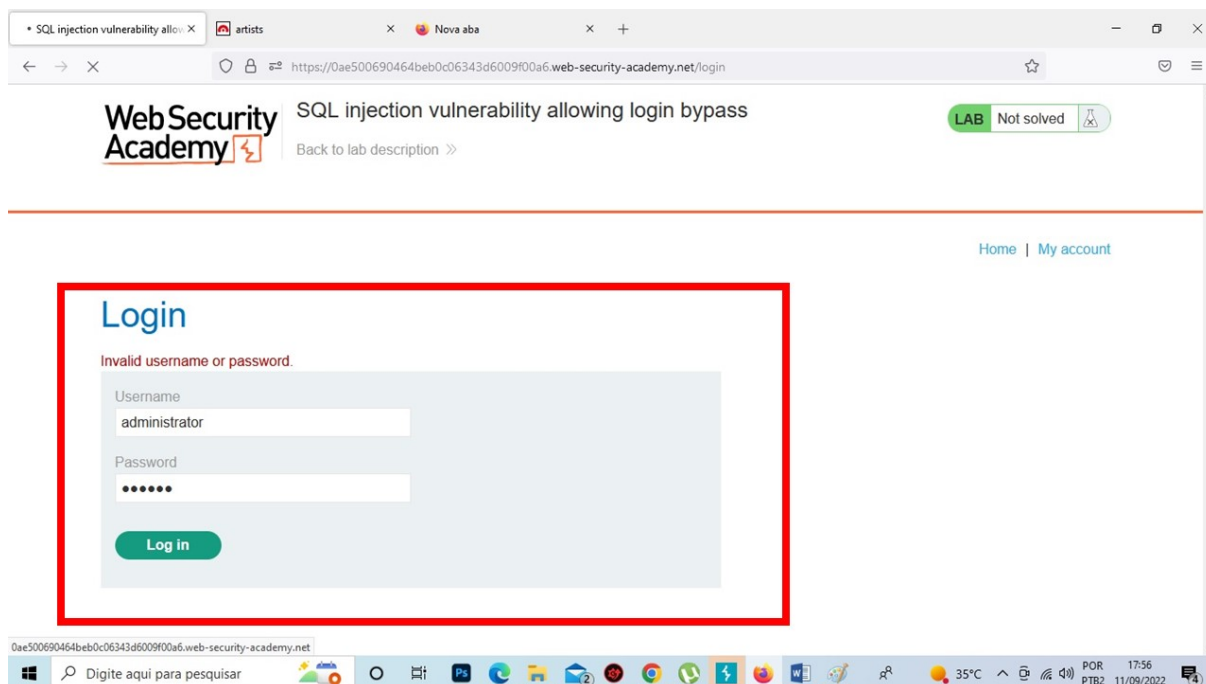
Figura 25 – Acesso ao laboratório Portswigger



Fonte: (PORTSWIGGER, 2022)

2º Passo: Neste exemplo foi inserido o nome de usuário Administrator e uma senha aleatória.

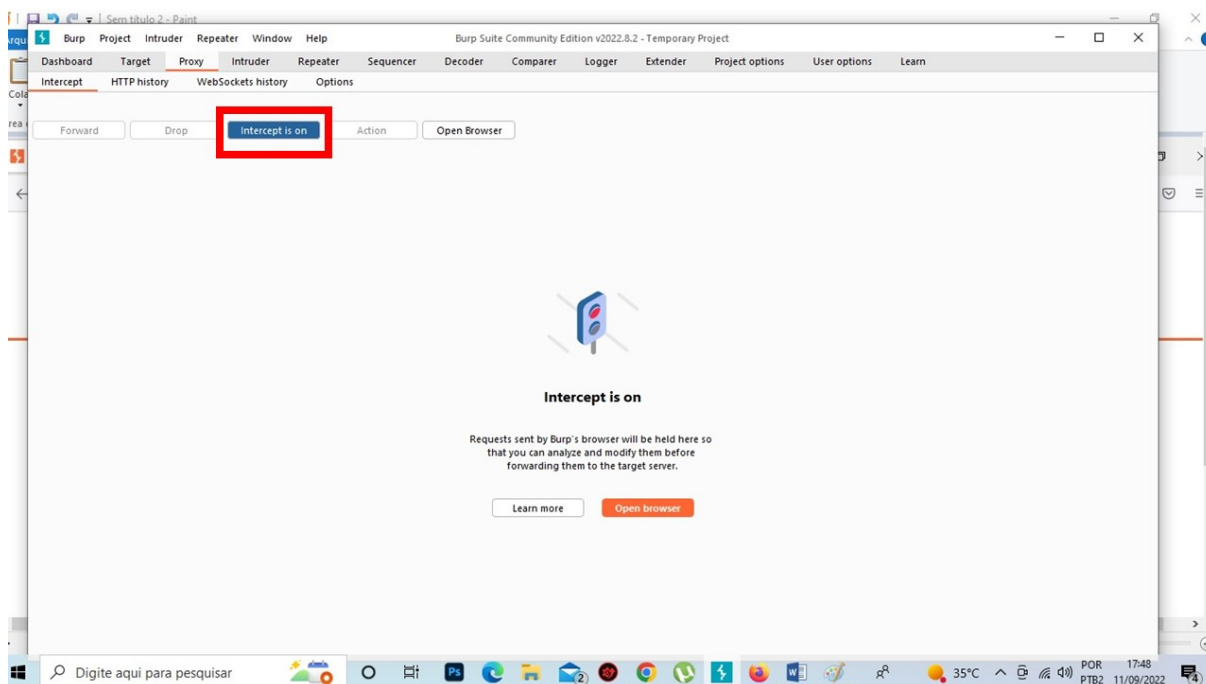
Figura 26 – Inseridos campos de usuário e senha no Laboratório Portswigger



Fonte: (PORTSWIGGER, 2022)

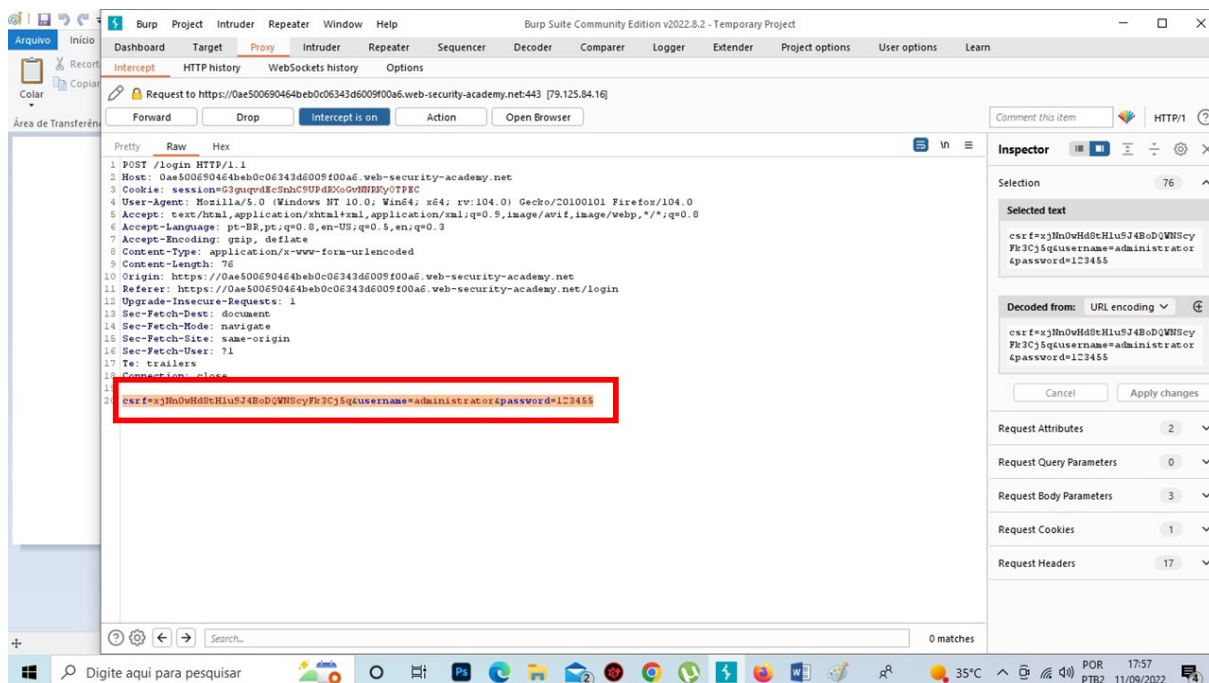
3º Passo Antes de clicar em *Log In*, foi iniciado o *BurpSuite*, e realizado a interceptação do site, após habilitar a interceptação do site pelo *BurpSuite* clica se em *Log In*.

Figura 27 – Tela de inteceptação do Burp Suite



Fonte: Próprio autor, 2022

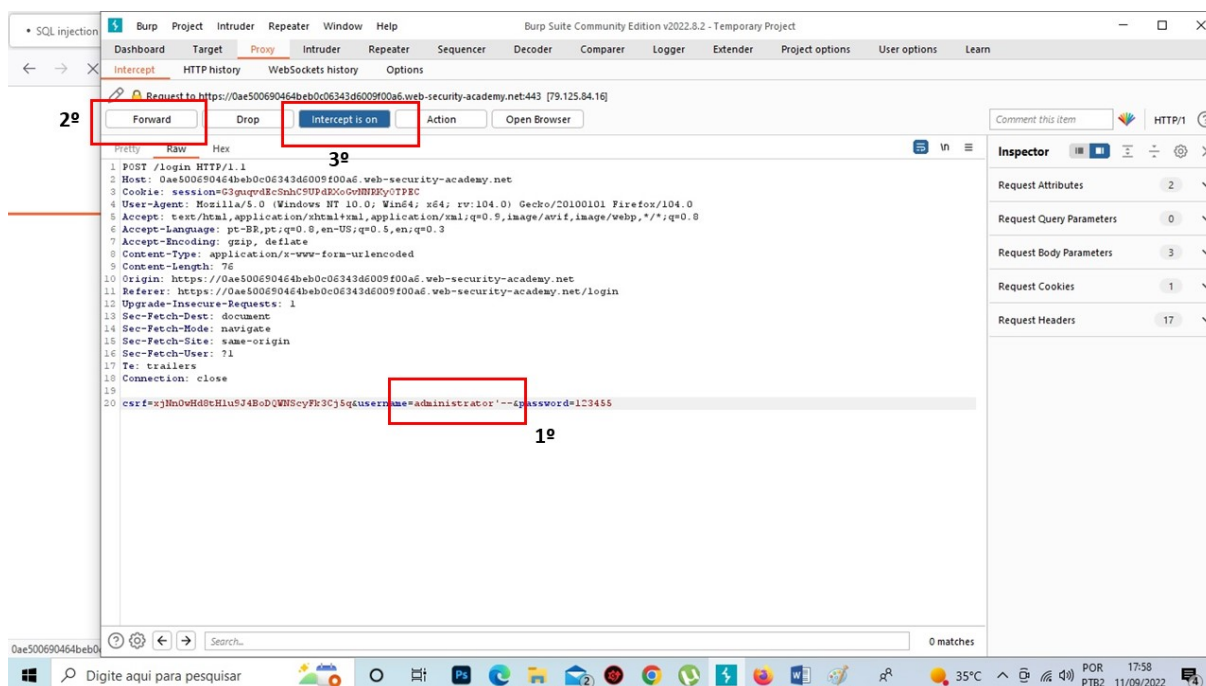
4º Passo: Localização dos campos de usuário e senha interceptados pelo *Burp Suite*.

Figura 28 – *Burp Suite*

Fonte: Próprio autor, 2022

5º Passo: Neste passo foi adicionado um comentário de linha ‘– após o nome de usuário administrator. Segundo o *site* (DEV MEDIA, 2007) o ataque será bem-sucedido porque a *string* ‘--’ faz com que o segundo comando após essa *string* apareça como um comentário sendo ignorado o restante do código, fazendo assim que se sabendo apenas o nome de usuário a senha será ignorada. Logo após adicionar o comentário de linha, clica-se em *Forward* e em seguida em *intercept is on*.

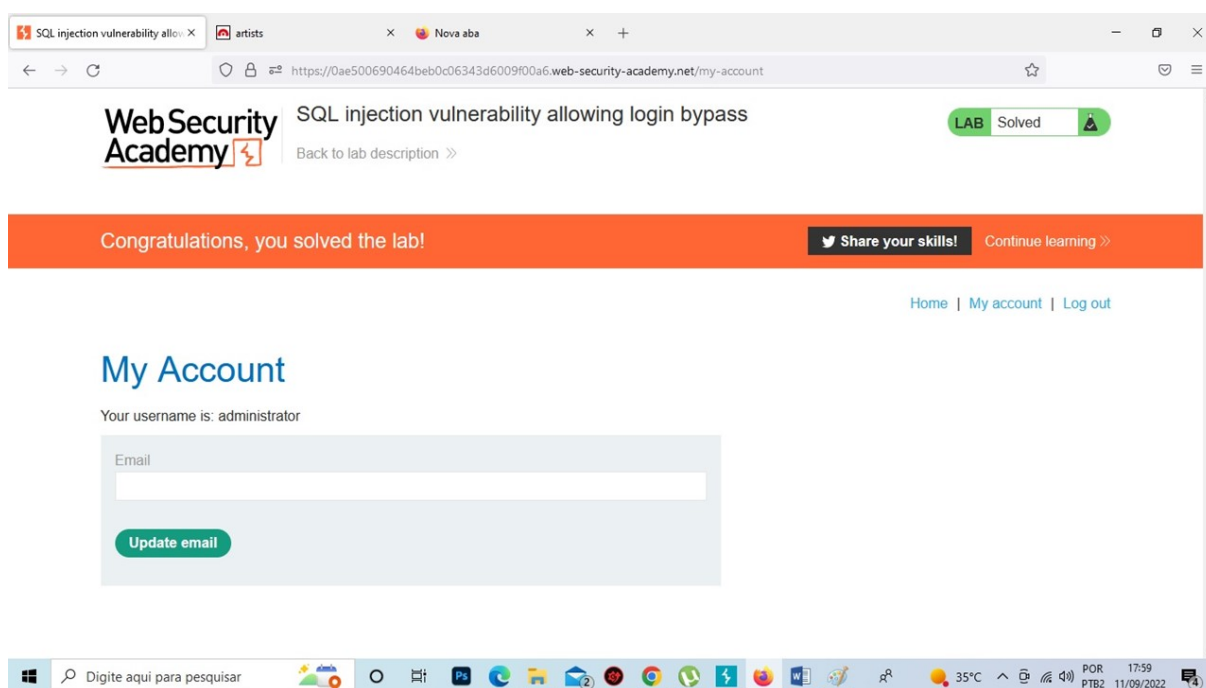
Figura 29 – Inserido código de injeção



Fonte: Próprio autor, 2022

6º Passo: Sendo assim conseguimos realizar o *login* no *site* sem a necessidade de digitar uma senha válida.

Figura 30 – Laboratório Portswigger acessado com sucesso



Fonte: (PORTSWIGGER, 2022)

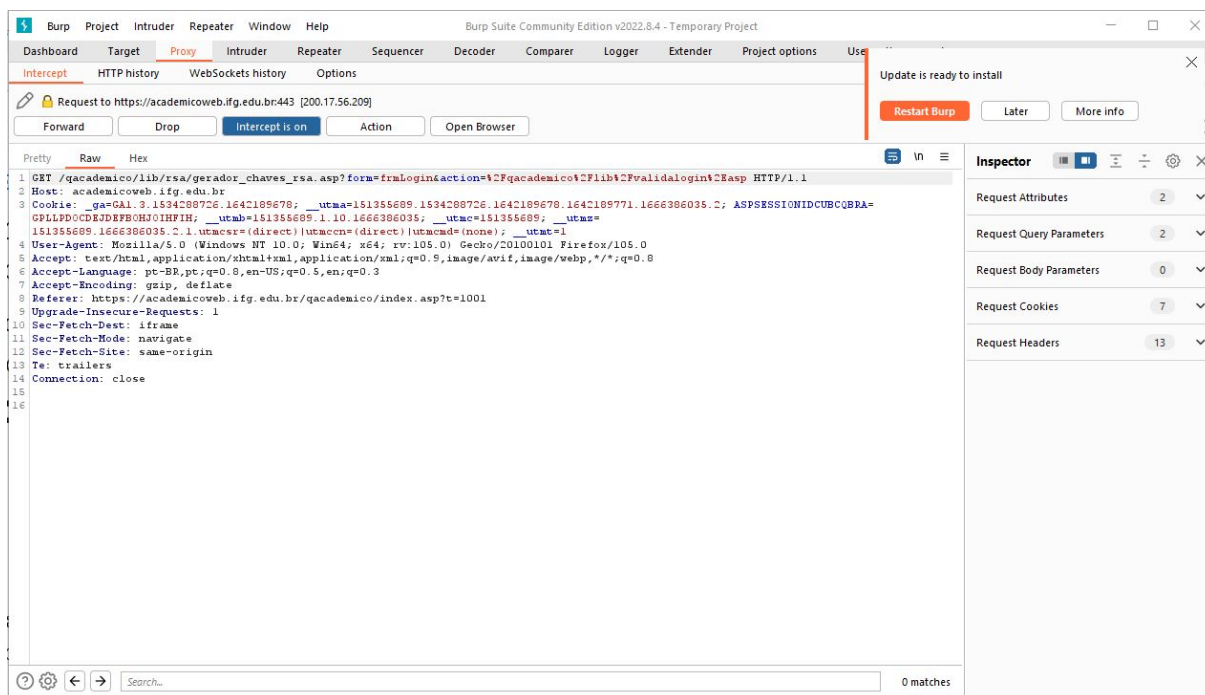
5.4 Implementação de Contramedidas

Através da análise prática das vulnerabilidades abordadas no Capítulo 5, forneceu uma ampla visão geral dos riscos e precauções mais importantes para lidar com qualquer tipo de site, seja ele governamental, institucional ou mesmo blogs pessoais. Nenhum aplicativo é totalmente seguro. Milhares de vulnerabilidades surgem todos os dias. As vulnerabilidades destacadas aqui são os pilares da maioria dos problemas de segurança em qualquer aplicação, de acordo com a metodologia *OWASP*.

Na Figura 31 realizamos um teste no site

<https://academicoweb.ifg.edu.br/qacademico/index.asp?t=1001> considerado seguro, fizemos a inteceptação dos pacotes através do *Burp Suite*, e obtivemos o retorno de que o site é seguro contra ataque de força bruta e injeção de códigos.

Figura 31 – Site seguro interceptado



Fonte: Próprio autor, 2022

6 CONCLUSÃO

Através da pesquisa realizada em segurança de aplicações *web* deparamos com grandes dificuldades em relação as metodologias de riscos, ameaças e criação dos ambientes de testes, e que foram proporcionados grandes desafios para resolução dos mesmos, revelou que, a maioria dos aplicativos da *web* possuem suas vulnerabilidades que podem causar danos enormes a uma organização ou indivíduo, dependendo da intenção do invasor de explorá-las. Portanto, boas práticas de programação que tratam de questões de segurança devem ser definidas, padronizadas e seguidas em todo o processo de desenvolvimento do sistema.

6.1 Análise dos resultados

Através das análises realizadas, foi possível identificar e classificar quais vulnerabilidades são mais exploradas de acordo com o *CVE* e o *OWASP*, para que medidas possam ser implementadas para preveni-las. Também podemos concluir que as vulnerabilidades mais comumente exploradas são principalmente devido à má codificação do sistema. Portanto, é importante que os desenvolvedores estejam cientes das vulnerabilidades de segurança e das estratégias de prevenção e incluam políticas de segurança no processo de desenvolvimento.

6.2 Trabalhos futuros

Como possível trabalho futuro pode-se sugerir estudos aprofundados em vulnerabilidades em aplicações *mobile*, redes *wireless*, implementação com base na metodologia de testes de penetração e exploração de vulnerabilidades *web*. O trabalho seria para atualizar o meio acadêmico em relação a segurança e auxiliar no desenvolvimento seguro com base no que os invasores mais estão tentando explorar atualmente.

Referências

- BANACH, Z. **O que é a falsificação de solicitação do lado do servidor (SSRF) e como você pode evitá-la?** 2021. Disponível em: <<https://www.invicti.com/blog/web-security/server-side-request-forgery-vulnerability-ssrf/>>. Acesso em: 05 de setembro de 2022. Citado 2 vezes nas páginas 17 e 22.
- BASTA, N. B. A.; BROWN, M. **Segurança de computadores e teste de invasão**. 2. ed. São Paulo: Cengage Learning, 2014. Citado 4 vezes nas páginas 7, 14, 18 e 19.
- BENETTI, T. **Segurança da informação confidencialidade, integridade e disponibilidade**. 2015. Disponível em: <<https://www.profissaista.com.br/2015/07/seguranca-da-informacao-confidencialidade-integridade-e-disponibilidade-cid>>. Acesso em: 15 de novembro de 2019. Citado na página 4.
- CECYBER. **Entenda a OWASP e a nova versão Top 10 Web Application Security Risks(2021)**. 2021. Disponível em: <<https://cecyber.com/owasp-top-10-web-application-security-risks-2021/>>. Acesso em: 19 de junho de 2022. Citado na página 21.
- CLOUDFLARE. **O que é um ataque de negação de serviço (DoS)?** 2022. Disponível em: <<https://www.cloudflare.com/pt-br/learning/ddos/glossary/denial-of-service/>>. Acesso em: 03 de outubro de 2022. Citado na página 8.
- COOPER, S. **Burp Suite Cheat Sheet**. 2022. Disponível em: <<https://www.comparitech.com/net-admin/burp-suite-cheat-sheet/>>. Acesso em: 2 de julho de 2022. Citado na página 29.
- DANTAS, M. **Segurança da informação: uma abordagem focada em gestão de riscos**. Olinda-PE: Livro Rápido, 2011. Citado na página 1.
- DETAILS, C. **CVE Details**. 2022. Disponível em: <<https://www.cvedetails.com/vulnerabilities-by-types.php>>. Acesso em: 20 de junho de 2022. Citado 4 vezes nas páginas 5, 6, 28 e 39.
- DEVMEDIA. **Injeção SQL**. 2007. Disponível em: <<https://www.devmedia.com.br/sql-injection/6102>>. Acesso em: 10 de julho de 2022. Citado na página 41.
- DURBANO, V. **Entenda o que é CVE (Common Vulnerabilities and Exposures)**. 2019. Disponível em: <<https://blog.ecoit.com.br/o-que-e-cve/>>. Acesso em: 19 de junho de 2022. Citado na página 5.
- HACKERSEC. **O que é um Payload e por que são usados por hackers?** 2018. Disponível em: <<https://hackersec.com/o-que-e-um-payload-e-por-que-sao-usados-por-hackers/>>. Acesso em: 25 de julho de 2022. Citado na página 34.
- HKCERT. **O OWASP Top 10-2021 já é lançado**. 2021. Disponível em: <<https://www.hkcert.org/blog/owasp-top-10-2021-is-now-released>>. Acesso em: 21 de agosto de 2022. Citado 3 vezes nas páginas 11, 13 e 16.

KASPERSKY. **O que é um ataque de força bruta?** 2022. Disponível em: <<https://www.kaspersky.com.br/resource-center/definitions/brute-force-attack>>. Acesso em: 28 de julho de 2022. Citado 2 vezes nas páginas 8 e 9.

KIRSTENS. **Cross Site Scripting**. 2017. Disponível em: <<https://owasp.org/www-community/attacks/xss/>>. Acesso em: 15 de fevereiro de 2021. Citado na página 20.

LIMA, D. de. **Vírus e antivírus o que é e como se proteger**. 2016. Disponível em: <<https://www.techtudo.com.br/dicas-e-tutoriais/noticia/2016/06/virus-e-antivirus-o-que-e-e-como-se-proteger.html>>. Acesso em: 16 de novembro de 2019. Citado na página 2.

LOPES, P. **ATAQUES DE INJEÇÃO**. 2018. Disponível em: <<https://periciacomputacional.com/ataques-de-injecao-injection-attacks/>>. Acesso em: 14 de fevereiro de 2021. Citado na página 20.

MACHADO, M. J. **Segurança da Informação: uma Visão Geral sobre as Soluções Adotadas em Ambientes Organizacionais**. Dissertação (Bacharelado) — Universidade Federal do Paraná, Curitiba, PR, 2012. Citado 2 vezes nas páginas 12 e 19.

MACORATTI, J. C. **Previna-se contra a Injeção SQL**. 2002. Disponível em: <https://www.macoratti.net/sql_inj.htm>. Acesso em: 10 de julho de 2022. Citado na página 9.

MARCONDES, J. S. **Conceito de segurança da informação organizacional. Gestão de segurança privada**. 2015. Disponível em: <<<https://gestaodesegurancaprivada.com.br/conceito-deseguranca-da-informacao-organizacional>>. Acesso em: 03 de novembro de 2019. Citado na página 1.

MOLINARO, D. **O que é um ataque de força bruta?** 2021. Disponível em: <<https://www.avast.com/pt-br/c-what-is-a-brute-force-attack>>. Acesso em: 02 de julho de 2022. Citado na página 8.

MORENO, D. **Pentest em Aplicações Web**. 1. ed. São Paulo: Editora Novatec, 2017. Citado 2 vezes nas páginas 34 e 39.

NAKAMURA, E. T.; GEUS, P. L. de. **Segurança de redes em ambientes cooperativos**. 1. ed. São Paulo: Novatec, 2007. Citado na página 18.

NOLETO, C. **Aplicações web: entenda o que são e como funcionam**. 2020. Disponível em: <<https://blog.betrybe.com/desenvolvimento-web/aplicacoes-web/>>. Acesso em: 15 de março de 2022. Citado na página 2.

OWASP. **OWASP Top 10:2021**. 2021. Disponível em: <<https://owasp.org/Top10/>>. Acesso em: 10 de julho de 2022. Citado 14 vezes nas páginas 10, 11, 12, 13, 14, 15, 16, 17, 19, 20, 21, 22, 25 e 39.

OWASPZAP, V. G. **Visao Geral OwaspZAP**. 2022. Disponível em: <<https://www.zaproxy.org/getting-started/>>. Acesso em: 12 de junho de 2022. Citado na página 26.

PORTSWIGGER. **O que é injeção de SQL?** 2021. Disponível em: <<https://portswigger.net/web-security/sql-injection>>. Acesso em: 02 de outubro de 2022. Citado na página 8.

PORTSWIGGER. **Falsificação de solicitação do lado do servidor (SSRF)**. 2022. Disponível em: <<https://portswigger.net/web-security/ssrf>>. Acesso em: 05 de setembro de 2022. Citado na página 22.

PORTSWIGGER. **Laboratório: vulnerabilidade de injeção de SQL que permite o bypass de login**. 2022. Disponível em: <<https://portswigger.net/web-security/sql-injection/lab-login-bypass>>. Acesso em: 11 de setembro de 2022. Citado 3 vezes nas páginas 39, 40 e 42.

REDACAO-CANALTECH. **O que é rootkit?** 2016. Disponível em: <<https://canaltech.com.br/seguranca/O-que-e-rootkit/#:~:text=Rootkit%20%C3%A9%20um%20software%2C%20na,com%20o%20in%C3%ADcio%20desse%20software>>. Acesso em: 15 de dezembro de 2020. Citado na página 7.

REGAN, J. **O que é um Cavalo de Troia? É malware ou vírus?** 2019. Disponível em: <<https://www.avg.com/pt/signal/what-is-a-trojan>>. Acesso em: 02 de janeiro de 2021. Citado na página 7.

RIBEIRO, C. da S. **Segurança da Informação: o desenvolvimento de uma política de segurança da informação em conformidade com a norma ABNT ISO/IEC 27002**. 35 p. Dissertação (Bacharelado) — FAIR Faculdades Integradas de Rondonópolis, Rondonópolis, 2016. Citado na página 18.

RIGUES, R. **Facebook atribui recente vazamento de dados de usuários a “scraping”**. 2021. Disponível em: <<https://olhardigital.com.br/2021/04/07/seguranca/facebook-atribui-recente-vazamento-de-dados-de-usuarios-a-scraping/>>. Acesso em: 23 de agosto de 2022. Citado na página 13.

SANDERS, C. **Compreendendo os ataques man-in-the-middle**. 2010. Disponível em: <<http://techgenix.com/Understanding-Man-in-the-Middle-Attacks-ARP-Part1/>>. Acesso em: 12 de janeiro de 2021. Citado na página 12.

SEGURANÇA, C. de. **Cartilha de segurança para internet**. 2017. Disponível em: <<https://cartilha.cert.br/malware/>>. Acesso em: 16 de janeiro de 2021. Citado 2 vezes nas páginas 6 e 7.

SEMOLA, M. **Gestão da Segurança da Informação - Uma Visão Executiva**. 2. ed. São Paulo: Elsevier, 2014. Citado na página 18.

SOLARWINDS. **IMPULSIONANDO A TI**. 2022. Disponível em: <<https://www.solarwinds.com/pt/>>. Acesso em: 02 de setembro de 2022. Citado na página 16.

SOUZA ADSON DIEGO LACERDA CAVALCANTE, A. I. B. d. S. J. C. d. S. Matheus Ronielli Alves de; SAMPAIO, R. R. F. **Ameaças e mecanismo de segurança da informação em um ambiente organizacional de planos de saúde**. 2018. Disponível em: <<http://periodicos.uern.br/index.php/ccontabil/article/download/3025/1609>>. Acesso em: 03 de novembro de 2019. Citado na página 1.

TURNER, J. **Violação da SolarWinds em dezembro de 2020: o que você precisa saber**. 2020. Disponível em: <<https://www.coalitioninc.com/blog/december-2020-solarwinds-breach-what-you>>. Acesso em: 02 de setembro de 2022. Citado na página 16.

VERAS, C. **Você sabe o que significa não repúdio**. 2018. Disponível em: <<https://solutiresponde.com.br/voce-sabe-o-que-significa-nao-repudio-descubra-neste-post/>>. Acesso em: 15 de agosto de 2021. Citado na página 4.

WESTPHALL, C. M.; RIGON, E. A. Modelo de avaliação da maturidade da segurança da informação: Revista eletrônica de sistemas de informação. v. 12, n. 1, 2013. Citado na página 4.