



Universidade de Brasília - UnB  
Faculdade UnB Gama - FGA  
Engenharia de Software

# **Avaliação de segurança de uma aplicação web: um estudo de caso**

**Autor: Carla Rocha Cangussú**  
**Orientador: Profa. Dra. Elaine Venson**

**Brasília, DF**  
**2024**



Carla Rocha Cangussú

## **Avaliação de segurança de uma aplicação web: um estudo de caso**

Monografia submetida ao curso de graduação em (Engenharia de Software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de Software).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Profa. Dra. Elaine Venson

Brasília, DF


2024

Carla Rocha Cangussú

## **Avaliação de segurança de uma aplicação web: um estudo de caso**

Monografia submetida ao curso de graduação em (Engenharia de Software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de Software).

Trabalho aprovado. Brasília, DF, 18 de setembro de 2024:

 Documento assinado digitalmente  
ELAINEVENSON  
Data: 30/09/2024 12:20:16-0300  
Verifique em <https://validar.it.gov.br>

---

**Profa. Dra. Elaine Venson**  
Orientador

---

**Prof.<sup>a</sup> Msc. Cristiane Soares Ramos**  
Convidado 1

---

**Prof. Dr. Sergio Antônio Andrade de Freitas**  
Convidado 2

Brasília, DF  
2024

# Agradecimentos

Gostaria de primeiramente agradecer a Deus pelas oportunidades que me foram concedidas, muito além do que um dia imaginei, incluindo o privilégio de ser aluna de uma das melhores universidades federais do país. Sem o Senhor, nada disso seria possível; foi Ele quem abriu todas as portas, me sustentou e me capacitou ao longo dessa desafiadora jornada que é a graduação em Engenharia de Software na UnB. Enfrentei muitos momentos em que minha fé foi testada, mas hoje posso afirmar que não o conheço apenas de ouvir falar, mas tenho um relacionamento íntimo e profundo. Este trabalho de conclusão de curso, assim como a minha trajetória acadêmica, são dedicados a honrar e glorificar o nome de Jesus, que é meu tudo, meu refúgio em tempos de aflição, o paz que excede todo o entendimento e verdadeira felicidade.

Realizar o sonho de cursar esta faculdade exigiu uma grande mudança em minha vida, sair do interior da Bahia e me estabelecer na capital do país. O maior desafio foi estar longe das pessoas que mais amo, minha mãe Lilione, meu pai Manuel Inácio, e minha irmã Manuela. Apesar da distância, em nenhum momento deixei de sentir o apoio incondicional de delas. Obrigada por todas as palavras de incentivo, conforto e por me fazerem sentir amada e cuidada, mesmo à distância. Eles acreditaram em mim, especialmente nos momentos em que eu mesma duvidava, e sem o amor e a confiança de vocês, eu jamais teria chegado tão longe.

Agradeço a todos os meus professores pelo tempo e dedicação ao ensino. Seus ensinamentos foram essenciais para minha formação como engenheira. Embora eu possa ter passado despercebida por muitos de vocês, saibam que todos vocês deixaram uma marca importante em minha jornada. Agradeço especialmente à minha orientadora, Elaine Venson, por acreditar na minha proposta de TCC, mesmo sendo um tema pouco abordado diretamente no curso. Sua presença constante ao longo do processo, seus ensinamentos sobre testes de segurança, e suas orientações valiosas me guiaram em cada etapa deste estudo. Muito obrigada! Também quero expressar minha gratidão à professora Cristiane Ramos e ao professor Sérgio Antônio, que me ouviram falar sobre assinaturas eletrônicas inúmeras vezes, tanto em suas disciplinas quanto no contexto deste trabalho. Sou grata por terem aceitado fazer parte da banca do meu TCC e pela maneira humana com que lecionam, o que torna o ambiente da graduação, muitas vezes hostil, mais acolhedor. Muito obrigada a ambos!

Por fim, mas não menos importante, agradeço à equipe do VALIDAR, com quem tive o privilégio de trabalhar por dois anos. Em especial, expresso minha gratidão ao meu supervisor e mentor, Ruy César Ramos Filho, que acreditou no meu potencial e me pro-

porcionou ensinamentos inestimáveis ao longo dessa jornada. Sua paixão por assinaturas eletrônicas e seu incentivo constante para que eu desse o meu melhor foram fundamentais para minha formação profissional. Sem você, eu não seria a profissional que sou hoje. Muito obrigada!

*“Que o nome de Deus seja louvado para sempre,  
pois dele são a sabedoria e o poder! É ele quem  
faz mudar os tempos e as estações; é ele quem  
põe os reis no poder e os derruba; é ele quem dá  
sabedoria aos sábios e inteligência aos inteligentes.  
(Bíblia Sagrada, Daniel 2, 20 e 21)*

# Resumo

A tecnologia a cada dia ganha mais espaço em nossas vidas, de modo que a internet passou a ser uma necessidade. Isto foi intensificado com a pandemia da COVID-19, principalmente nas áreas de comércio, educação e saúde. No setor da saúde foi adotada a prática da telemedicina, isto é, a prestação de serviços de saúde de maneira remota com auxílio da tecnologia. Desta maneira, o uso de documentos digitais assinados eletronicamente cresceu, exemplo destes são atestados, relatórios e prescrições eletrônicas. Com isso, o Instituto Nacional de Tecnologia da Informação (ITI) criou um serviço web de validação de assinaturas eletrônicas, o VALIDAR, para garantir a autoria e integridade das assinaturas e documentos digitais. Mas, assim como aumentou o uso da internet, também houve o crescimento de crimes cibernéticos como ataques e roubo de dados, trazendo à tona a necessidade de desenvolver softwares seguros. O presente estudo tem como objetivo realizar a avaliação da segurança do *VALIDAR - Serviço de validação de assinaturas eletrônicas* empregando a metodologia de estudo de caso único com base no protocolo de Brereton, e utilizando técnicas de modelagem de ameaças, além de testes de segurança estáticos e dinâmicos. A análise revelou vulnerabilidades críticas, principalmente relacionadas à disponibilidade do serviço, como riscos de ataques de negação de serviço (DoS) e à codificação de endereços IP no código-fonte. Embora essas vulnerabilidades não comprometam a integridade das validações, elas afetam a performance e a disponibilidade do serviço. Conclui-se que é possível mitigar esses riscos por meio da implementação de *rate limiting*, balanceamento de carga e melhores práticas de configuração, melhorando a resiliência e a segurança do VALIDAR.

**Palavras-chaves:** Assinaturas eletrônicas; Segurança de software; Avaliação de segurança.

# Lista de ilustrações

Figura 1 – Ciclo de processos do Microsoft TMT (Microsoft, 2023b) . . . . .	26
Figura 2 – Posição do ZAP (OWASP, 2023) . . . . .	31
Figura 3 – Posição do Zap quando há <i>proxy</i> (OWASP, 2023) . . . . .	32
Figura 4 – Representação da API do VALIDAR . . . . .	39
Figura 5 – Pilares da Validade Jurídica (ITI, a) . . . . .	40
Figura 6 – Representação da antiga arquitetura do VALIDAR . . . . .	42
Figura 7 – Representação da arquitetura atual do VALIDAR . . . . .	42
Figura 8 – Diagrama simplificado de verificação de assinatura digital (ITI, 2021) .	44
Figura 9 – Diagrama simplificado da arquitetura do sistema VALIDAR . . . . .	47
Figura 10 – Janela de propriedades da ameaça . . . . .	47
Figura 11 – Gráfico de contagem de ameaças por categoria . . . . .	53
Figura 12 – Gráfico de contagem de interações por tipo . . . . .	54
Figura 13 – Gráfico de contagem de ameaças por categoria . . . . .	54
Figura 14 – Gráfico de contagem de ameaças por categoria e por severidade . . . .	55
Figura 15 – Resultado do SAST para o <i>front-end</i> do Validar. . . . .	58
Figura 16 – Resultado do SAST para o <i>middleware</i> do Validar. . . . .	58
Figura 17 – Resultado do SAST para a API do Validar. . . . .	59
Figura 18 – Quantidade de pontos de acesso de segurança por parte testada. . . . .	60
Figura 19 – Primeiro ponto crítico de segurança encontrado da API. . . . .	61
Figura 20 – Segundo crítico de segurança encontrado da API. . . . .	62
Figura 21 – Classificação de severidade dos pontos de acesso de segurança encontrados no <b>middleware</b> . . . . .	63
Figura 22 – Primeiro ponto de acesso de segurança encontrados no <b>middleware</b> . .	64
Figura 23 – Segundo ponto de acesso de segurança encontrados no <b>middleware</b> . .	65
Figura 24 – Terceiro ponto de acesso de segurança encontrados no <b>middleware</b> . .	66
Figura 25 – Quarto ponto crítico de segurança encontrados no <b>middleware</b> . . . .	67
Figura 26 – Quinto ponto crítico de segurança encontrados no <b>middleware</b> . . . .	68
Figura 27 – Sexto ponto crítico de segurança encontrados no <b>middleware</b> . . . . .	69
Figura 28 – Sétimo ponto crítico de segurança encontrados no <b>middleware</b> . . . .	69
Figura 29 – Oitavo ponto crítico de segurança encontrados no <b>middleware</b> . . . .	70
Figura 30 – Nono ponto crítico de segurança encontrados no <b>middleware</b> . . . . .	71
Figura 31 – Classificação de severidade dos pontos críticos de segurança encontrados no <b>front-end</b> . . . . .	72
Figura 32 – Pontos críticos da categoria de Injeção de código (RCE) encontrados <i>front-end</i> . . . . .	73



Figura 33 – Pontos de acesso da categoria de Injeção de código (RCE) encontrados <i>front-end</i> . . . . .	74
Figura 34 – Pontos críticos da categoria de Injeção de código (RCE) encontrados <i>front-end</i> . . . . .	75
Figura 35 – Ponto de acesso de segurança sobre permissão encontrado <i>front-end</i> . .	75
Figura 36 – Pontos críticos de segurança relacionados à verificação de integridade de recursos encontrados no <i>front-end</i> . . . . .	76
Figura 37 – Pontos críticos de segurança relacionados à ausência do parâmetro <i>no- opener</i> encontrados no <i>front-end</i> . . . . .	76
Figura 38 – Alertas identificados no DAST . . . . .	78
Figura 39 – Vulnerabilidade de alto risco identificada pelo ZAP . . . . .	79
Figura 40 – Pontos críticos de segurança relacionados à ausência . . . . .	80
Figura 41 – Comparação dos resultados obtidos nos três métodos utilizados no es- tudo de caso . . . . .	87

# Lista de tabelas

Tabela 1 – Pilares de segurança e ameaças, segundo o framework STRIDE (HEWKO, 2021) . . . . .	28
Tabela 2 – Tabela para cálculo do nível do risco . . . . .	28
Tabela 3 – Priorização realizada com o DREAD . . . . .	49
Tabela 4 – Dados coletados na modelagem de ameaças . . . . .	55
Tabela 5 – Dados coletados na modelagem de ameaças . . . . .	55
Tabela 6 – Análise conjunta dos resultados . . . . .	86
Tabela 7 – Dados coletados na modelagem de ameaças . . . . .	101

# Lista de abreviaturas e siglas

AC	Autoridade Certificadora
API	Application Programming Interface
APISIX	API Gateway
CDNs	Content Delivery Networks
CFM	Conselho Federal de Medicina
CORS	Cross-Origin Resource Sharing
CRF	Conselho Regional de Farmácia
CRO	Conselho Regional de Odontologia
CSRF	Cross-Site Request Forgery
CSS	Cascading Style Sheets
CSP	Content Security Policy
DAST	Dynamic Application Security Testing
DOC-ICP-15	Documento do ICP-Brasil
DoS	Denial of Service
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
httpd	HTTP Daemon
HTTPS	Hypertext Transfer Protocol Secure
HSTS	HTTP Strict Transport Security
IIS	Internet Information Services
ICP-Brasil	Infraestrutura de Chaves Públicas Brasileira
INE	Departamento de Informática e de Estatística
ITI	Instituto de Tecnologia da Informação

JSON	JavaScript Object Notation
LabSEC	Laboratório de Segurança em Computação
MFA	Multi-Factor Authentication
MERCOSUL	Mercado Comum do Sul
PBAD	Padrão Brasileiro de Assinatura Digital
SAST	Static Application Security Testing
SDL	Microsoft Security Development Lifecycle
SSL	Secure Sockets Layer
SQL	Structured Query Language
TMT	Microsoft Threat Modeling Tool
UFSC	Universidade Federal de Santa Catarina
XSRF	Cross-Site Request Forgery (ou CSRF)
XSS	Cross-Site Scripting

# Sumário

<b>1</b>	<b>INTRODUÇÃO</b>	<b>15</b>
1.1	Problema	16
1.2	Objetivo	17
1.3	Metodologia	17
1.4	Organização do documento	18
<b>2</b>	<b>REFERENCIAL TEÓRICO</b>	<b>20</b>
2.1	Segurança de Software	20
2.2	Vulnerabilidade	21
2.3	Ciclo de vida de desenvolvimento de software seguro	21
2.3.1	Microsoft Security Development Lifecycle (SDL)	22
2.4	Modelagem de ameaças	25
2.4.1	Microsoft Threat Modeling Tool (TMT)	26
2.4.2	STRIDE	27
2.4.3	DREAD	28
2.5	Testes de segurança estático	29
2.5.1	SonarQube	30
2.6	Testes de segurança dinâmicos	30
2.6.1	OWASP Zap	31
<b>3</b>	<b>METODOLOGIA</b>	<b>33</b>
3.1	Protocolo de Brereton	33
3.2	Aplicação do protocolo ao estudo	35
<b>4</b>	<b>ANÁLISE DO VALIDAR</b>	<b>38</b>
4.1	Assinaturas eletrônicas	39
4.1.1	Validade jurídica	40
4.1.2	Classificação das assinaturas eletrônicas	40
4.2	Aspectos técnicos	41
4.2.1	Arquitetura	41
4.2.2	Tecnologias	43
4.3	O processo de validação	44
<b>5</b>	<b>RESULTADOS DA MODELAGEM DE AMEAÇAS</b>	<b>46</b>
5.1	Passo-a-passo da coleta de dados	46
5.2	Dados coletados da modelagem de ameaças	49

5.2.1	Dados do DREAD . . . . .	49
<b>5.3</b>	<b>Análise dos dados obtidos . . . . .</b>	<b>52</b>
<b>6</b>	<b>RESULTADOS DO TESTE ESTÁTICO DE SEGURANÇA . . . . .</b>	<b>56</b>
<b>6.1</b>	<b>Passo-a-passo da coleta de dados . . . . .</b>	<b>56</b>
<b>6.2</b>	<b>Dados coletados do teste de segurança estático . . . . .</b>	<b>57</b>
6.2.1	Dados do <i>front-end</i> do Validar . . . . .	58
6.2.2	Dados do <i>middleware</i> do Validar . . . . .	59
6.2.3	Dados da API do Validar . . . . .	59
<b>6.3</b>	<b>Análise dos dados obtidos . . . . .</b>	<b>59</b>
6.3.1	Análise dos dados da API do Validar . . . . .	60
6.3.2	Análise dos dados do <i>middleware</i> do Validar . . . . .	62
6.3.2.1	Ponto crítico de segurança de negação de serviço (DoS) encontrados no <i>middleware</i>	65
6.3.2.2	Ponto crítico de segurança sobre permissão encontrados no <i>middleware</i> . . . . .	66
6.3.2.3	Ponto crítico de segurança sobre criptografia de dados confidenciais encontrados no <i>middleware</i> . . . . .	66
6.3.2.4	Pontos críticos de segurança de configuração insegura encontrados no <i>middleware</i>	67
6.3.2.5	Outros pontos críticos de segurança de encontrados no <i>middleware</i> . . . . .	68
6.3.3	Análise dos dados do <i>front-end</i> do Validar . . . . .	71
6.3.3.1	Pontos críticos da categoria de Injeção de código (RCE) encontrados <i>front-end</i> .	72
6.3.3.2	Pontos de acesso da categoria de Negação de serviço (DoS) encontrados <i>front-end</i>	73
6.3.3.3	Ponto crítico de segurança sobre permissão encontrado <i>front-end</i> . . . . .	73
6.3.4	Pontos críticos de segurança do tipo Outros . . . . .	74
<b>7</b>	<b>RESULTADOS DO TESTE DE SEGURANÇA DINÂMICO . . . . .</b>	<b>77</b>
<b>7.1</b>	<b>Passo-a-passo da coleta de dados . . . . .</b>	<b>77</b>
<b>7.2</b>	<b>Dados coletados do teste de segurança dinâmico . . . . .</b>	<b>77</b>
7.2.1	Alertas de alto risco . . . . .	78
7.2.2	Alertas de risco médio . . . . .	79
7.2.2.1	Ausência de tokens Anti-CSRF . . . . .	79
7.2.2.2	Configuração Incorreta Entre Domínios . . . . .	80
7.2.2.3	Ausência do cabeçalho de <i>Content Security Policy (CSP)</i> . . . . .	80
7.2.2.4	Falta de um cabeçalho de proteção contra <i>clickjacking</i> . . . . .	81
7.2.2.5	Biblioteca JS vulnerável . . . . .	81
7.2.3	Alertas de risco baixo . . . . .	81
7.2.3.1	Inclusão de arquivo fonte <i>JavaScript</i> entre domínios . . . . .	82
7.2.3.2	Divulgação de Data e Hora - Unix . . . . .	82
7.2.3.3	Servidor vaza informações de versão por meio do campo de cabeçalho de resposta HTTP "Servidor" . . . . .	82
7.2.3.4	<i>Strict-Transport-Security Header Not Set</i> . . . . .	83

7.2.3.5	<i>X-Content-Type-Options Header Missing</i> . . . . .	83
7.2.4	Alertas de caráter informacional . . . . .	84
7.2.4.1	Divulgação de Informações - Comentários Suspeitos . . . . .	84
7.2.4.2	<i>Modern Web Application</i> . . . . .	84
7.2.4.3	<i>Re-examine Cache-control Directives</i> . . . . .	84
7.2.4.4	<i>User Controllable HTML Element Attribute (Potential XSS)</i> . . . . .	85
<b>8</b>	<b>ANÁLISE CONJUNTA DOS RESULTADOS DAS TRÊS TÉCNICAS</b>	<b>86</b>
<b>8.1</b>	<b>Dados coletados</b> . . . . .	<b>86</b>
<b>8.2</b>	<b>Análise conjunta dos dados</b> . . . . .	<b>87</b>
<b>8.3</b>	<b>Propostas de mitigação</b> . . . . .	<b>89</b>
8.3.1	Falsificação da Entidade Externa do Usuário Humano . . . . .	90
8.3.2	Falha ou parada potencial do processo para API . . . . .	90
8.3.3	Interrupção Potencial do Fluxo de Dados HTTPS . . . . .	92
<b>9</b>	<b>CONCLUSÃO</b> . . . . .	<b>93</b>
	<b>REFERÊNCIAS</b> . . . . .	<b>95</b>
	<b>APÊNDICES</b>	<b>100</b>
	<b>APÊNDICE A – PRIMEIRO APÊNDICE</b> . . . . .	<b>101</b>
	<b>ANEXO A – PRIMEIRO ANEXO</b> . . . . .	<b>111</b>
	<b>ANEXO B – SEGUNDO ANEXO</b> . . . . .	<b>114</b>

# 1 Introdução

O mundo está em meio à quarta revolução industrial. Isso significa, “um crescimento exponencial da capacidade de computação e combinação de tecnologias físicas, digitais e biológicas” (MAGALHÃES; VENDRAMINI, 2018). Algumas das tecnologias marcantes desta revolução são a inteligência artificial, a robótica, a biotecnologia, a nanotecnologia, o *blockchain*, a internet das coisas (IoT) e a impressão em três dimensões (3D) (SCHWAB, 2019). Estas tecnologias, antes tão distantes de nós, hoje fazem parte do nosso dia a dia, um bom exemplo disto é o chatGPT, que é uma inteligência artificial no formato *chatbot* que foi lançado em 2022.

A pandemia do COVID-19 contribuiu para o aceleração da adoção ampla dessas tecnologias em diversas áreas, tais como comércio, educação e saúde. Segundo o SEBRAE (2021), em sua 13ª edição da pesquisa “O impacto do Coronavírus nos pequenos negócios”, 55% das empresas modificaram sua forma de funcionamento devido a crise. Grande parte desta mudança ocorreu no modo de comercialização, 74% desses pequenos negócios vende utilizando redes sociais, aplicativos ou internet. Este valor teve uma alta de 15% comparado à 3ª edição da mesma pesquisa realizada em 2020.

Outra área que foi fortemente impactada pela pandemia e precisou da ajuda da tecnologia para se adaptar à nova realidade foi a educação. Uma vez que as aulas presenciais foram suspensas e substituídas por atividades não presenciais ancoradas em meios digitais enquanto durou a situação de pandemia do Coronavírus (COVID-19) (VIEIRA; SILVA, 2020). Os gestores educacionais, professores e alunos se viram obrigados a adotar um novo modelo educacional sustentado pelas tecnologias digitais e pautado nas metodologias da educação online (VIEIRA; SILVA, 2020).

Assim como as áreas citadas acima foram impactadas pela pandemia e buscaram soluções nas tecnologias, também foi com a saúde. Uma das principais estratégias neste setor foi o uso da telemedicina. Isto é, a prestação de serviços de saúde de maneira remota, dispondo-se do uso de tecnologias de informação e comunicação por profissionais da área com vistas à prevenção, ao tratamento ou ao diagnóstico de doenças (LIMA et al., 2022). Com a adoção da telemedicina houve um aumento do uso de documentos digitais na área da saúde, exemplos desses são atestados, relatórios e prescrições eletrônicas.

Foi neste contexto que o Instituto Nacional de Tecnologia da Informação (ITI) desenvolveu o *Validador de documentos digitais de saúde*, aplicação que valida se as assinaturas digitais pertencem a profissionais de saúde. Esta aplicação foi desenvolvida a partir do *Verificador de conformidade*, ferramenta que verifica se as assinaturas digitais estão em conformidade com o padrão estabelecido pela ICP-Brasil (Infraestrutura de



Chaves Públicas Brasileira).

O objeto de estudo do presente documento, o *VALIDAR - Serviço de validação de assinaturas eletrônicas*, é a junção destas duas aplicações, o Verificador e o Validador de documentos digitais de saúde, que realiza tanto a verificação de conformidade da assinatura com o padrão, quanto a validação se a mesma pertence a um profissional de saúde.

Com o isolamento causado pela Pandemia da COVID-19, as pessoas passaram a estar cada vez mais conectadas. A internet se tornou uma necessidade, porém, mesmo diante das coisas boas que ela permite que sejam realizadas, existem pessoas que recorrem a métodos ilícitos buscando o ganho próprio sobre os outros (JUNIOR; JUSTINO, 2022). Com tantos dados circulando na internet, muitos de maneira desprotegida, houve o aumento do índice de ataques, fraudes, invasão de privacidade, roubo e exclusão de dados, também conhecidos como crimes cibernéticos. Segundo Junior e Justino (2022), entre Fevereiro de 2019 e Fevereiro de 2020, os números de casos de ciberataques cresceram em número recorde de 308,17%.

O crescimento no índice de crimes cibernéticos reflete a necessidade de se ampliar o uso das práticas de segurança no desenvolvimento e manutenção de software. Devido a isso, as áreas de Segurança de Software e Desenvolvimento de Software Seguro, vem ganhando mais atenção. Elas consideram que existem aspectos que não são apenas conceitos ligados à segurança de forma geral, mas sim ao processo de desenvolvimento de software, em que o desenvolvedor precisa deixar de agir de forma reativa aos problemas de segurança, passando a construir software seguro desde a sua concepção (CALIL, 2023).

A busca pela criação do software seguro desde o seu primeiro momento originou o conceito "Ciclo de vida de desenvolvimento de software seguro". A *Microsoft* foi uma das pioneiras no uso deste conceito com o *Microsoft Security Development Lifecycle (SDL)*. O foco do SDL é aplicar práticas de segurança durante todo o processo de desenvolvimento do software e não só final como muitas vezes ocorre (Laurie Williams, 2021). Algumas das práticas presentes neste ciclo de vida são o levantamento de requisitos de segurança, modelagem de ameaças, o uso de criptografia, aplicação de teste de penetração e teste de segurança estático e dinâmico (Microsoft, 2023c). Desta maneira, as vulnerabilidades e falhas de segurança são tratadas desde sua origem, o que impede sua progressão e minimiza o retrabalho ao final do desenvolvimento. Como resultado, o produto final disponibilizado ao mercado é mais seguro, dificultando os ciberataques.

## 1.1 Problema

Assim como todo software está sujeito a pessoas maliciosas, o VALIDAR também está. A presença de brechas de segurança neste serviço pode vir a lesar usuários de bem,

caso sejam exploradas por outros mal intencionados, pois os resultados apresentados pelo VALIDAR muitas vezes são utilizados em transações comerciais, em processos jurídicos e na compra de medicamentos, por exemplo. Para evitar que documentos fraudulentos, seja com assinatura falsa ou com a alteração do conteúdo do documento, estejam em circulação se passando como válidos é necessários mitigar as brechas de seguranças deste serviço, se houver. Para minimizar este tipo de ocorrência é fundamental realizar uma avaliação de segurança, para que suas vulnerabilidades sejam diagnosticadas, priorizadas e resolvidas.

## 1.2 Objetivo

O presente trabalho objetiva realizar uma avaliação de segurança no VALIDAR - serviço de validação de assinaturas eletrônicas, prestado pelo ITI. Esta avaliação envolverá os seguintes objetivos específicos:

- Mapear os riscos de segurança do VALIDAR, por meio da modelagem de ameaças.
- Detectar as falhas de segurança no VALIDAR, a partir das abordagens de teste de segurança estático e dinâmico, também conhecidas pelas abreviação dos termos em inglês, SAST e DAST, respectivamente.
- Identificar os principais fatores que contribuem para a presença destas vulnerabilidades.
- Realizar recomendações para mitigar as vulnerabilidades encontradas.

## 1.3 Metodologia

A metodologia aplicada no presente trabalho será de estudo de caso, por meio do protocolo de [Brereton et al. \(2008\)](#), que consiste em uma descrição da sequência de eventos observados e análise dos dados coletados a partir destes eventos.

Para a coleta de dados serão utilizados como recurso a aplicação de algumas das práticas mencionadas no *Microsoft Security Development Lifecycle (SLD)*, ([Microsoft, 2023c](#)). Sendo essas, a técnicas modelagem de ameaças por meio de modelo *STRIDE da Microsoft* ([Microsoft, 2023a](#)) e a ferramenta *Microsoft Threat Modeling Tool* ([Microsoft, 2023b](#)), para levantamento e priorização dos risco de segurança. A análise dinâmica de testes de segurança, com o auxílio da ferramenta OWASP Zap ([OWASP, 2023](#)), e a análise estática de testes de segurança, por meio da ferramenta *SonarQube* ([SONAR, 2023](#)).

Os resultados obtidos por ambas das técnicas de teste aplicadas serão analisados tanto de forma individual como em conjunto, pois segundo ([ELDER et al., 2022](#)), estas

são complementares. Como estas informações será possível entender melhor cada vulnerabilidade e assim propor meios de mitigá-las, para que documentos fraudulentos não estejam em circulação e venham lesar cidadãos de bem.

Espera-se que com a aplicação destas técnicas, haja o aumento da segurança do software antes da tentativa de burlar os resultados do serviço podendo acarretar em danos ou mesmo crimes, o que por sua vez trará um aumento da confiabilidade no serviço que visa garantir a integridade e autenticidades de documentos digitais e suas assinaturas eletrônicas.

## 1.4 Organização do documento

O presente trabalho é composto por seis capítulos e possui a seguinte organização:

Capítulo 1 - Introdução: apresenta o contexto em que o estudo está inserido e o objeto de estudo, de modo que introduz a problemática e deixa claro seu objetivo.

Capítulo 2 - Referencial Teórico: traz ao conhecimento do leitor os conceitos base para o desenvolvimento do tema alvo do trabalho, uma vez que estes serão utilizados durante todo o decorrer do estudo, com a devida análise dos trabalhos relevantes, encontrados nas pesquisas e referências bibliográficas sobre o tema.

Capítulo 3 - Metodologia: discorre sobre as técnicas e procedimentos metodológicos utilizados no trabalho para o planejamento deste, a coleta de dados, as escolhas na construção dos artefatos, os cuidados com a análise dos dados e também detalhes sobre os testes a serem realizados. Além disso, apresenta o cronograma completo do estudo de caso, inclusive próximos passos para a síntese do TCC 2.

Capítulo 4 - Análise do VALIDAR: apresenta o serviço VALIDAR e o contexto que está inserido, tecnologias utilizadas e como funciona o processo de validação das assinaturas eletrônicas.

Capítulo 5 - Resultados da Modelagem de Ameaças: expõe todo o procedimento realizado para a coleta de dados com esta técnica, com o auxílio da ferramenta *Microsoft Thread Modeling Tool* em conjuntos com os *frameworks da Microsoft* STRIDE e DREAD, os dados obtidos e por fim a análise propriamente dos resultados encontrados.

Capítulo 6 - Resultados do Teste Estático de Segurança: descreve o passo-a-passo executado neste estudo de caso para coletar os dados do teste estático de segurança (SAST), usando o *SonarQube*. Na sequência, os dados coletados são analisados e dissecados empregando tabelas e gráficos.

Capítulo 7 - Resultados do Teste de Segurança Dinâmico: traz ao conhecimento do leitor o passo a passo realizado para coletar os dados do teste dinâmico de segurança (DAST) utilizando o OWASP Zap. Além disso, os dados coletados serão minuciosamente

registrados, analisados e examinados.

Capítulo 8 - Análise conjunta dos resultados das três técnicas: realiza a análise, a comparação e a discussão dos resultados obtidos a partir das três técnicas avaliadas: modelagem de ameaças, SAST e DAST. Também é neste capítulo que são apresentadas as sugestões de mitigação para as vulnerabilidades identificadas, não são considerados os falsos positivos identificados.

Capítulo 9 - Conclusão: apresentas as conclusões obtidas durante o estudo de caso, respondendo as questões que nortearam este trabalho.

## 2 Referencial teórico

Nesta seção serão abordados os seguintes temas, Segurança de software, Ciclo de vida do desenvolvimento do Software Seguro, *Microsoft Security Development Lifecycle*, popularmente conhecido com SDL. Além disso, serão tratadas as técnicas de modelagem de ameaças, *Static Analysis Security Testing (SAST)*, ou teste estático de segurança de aplicações, e *Dynamic Application Security Testing*, ou teste dinâmico de segurança de aplicações.

### 2.1 Segurança de Software

Com o avanço veloz da tecnologia e o crescimento de sua dependência, tanto por parte da população como pelas instituições públicas e privadas, a segurança no software passou a ser um requisito fundamental para os fornecedores de software. Isto acontece, pois é uma necessidade crítica proteger e preservar a confiança do ambiente computacional (Júnior; DOS, 2013).

Um sistema de software é seguro se satisfaz um objetivo de segurança especificado ou implícito. Este objetivo de segurança deve considerar requisitos de confidencialidade, integridade e disponibilidade para os dados e funcionalidade do sistema (PIESSENS, 2021). Embora, não exista um padrão de desenvolvimento de software seguro, pois cada aplicação é única, com propósito particular e por isso possui vulnerabilidades específicas, a adoção desses requisitos pode contribuir para a segurança.

Segundo Santos et al. (2019), a confidencialidade garante a proteção de toda a informação sensível e só permite seu acesso por pessoas autorizadas. O requisito da integridade é responsável por assegurar que as informações não sofram alterações e sejam mantidas exatamente como foram disponibilizadas pelo usuário. Já a disponibilidade, é requisito que torna as informações geradas ou adquiridas por indivíduos ou instituições disponíveis a qualquer momento para os usuários autorizados.

Os diversos ambientes que dependem da informação e de seus ativos podem também contar com sua própria política de segurança da informação, contendo normas e regras que devem ser seguidas de acordo com a necessidade e anseio de proteção (Júnior; DOS, 2013).

## 2.2 Vulnerabilidade

No contexto da segurança do software, as vulnerabilidades são falhas ou descuidos específicos em um software que permitem que os atacantes façam algo malicioso - expor ou alterar informações confidenciais, interromper ou destruir um sistema ou assumir o controle de um sistema ou programa de computador (DOWD; MCDONALD; SCHUH, 2006).

Normalmente, vulnerabilidades de segurança são *bugs* que trazem uma surpresa oculta extra: um usuário mal-intencionado que pode aproveitá-los para lançar ataques contra o software e os sistemas de suporte. Quase todas as vulnerabilidades de segurança são *bugs* de software, mas apenas alguns erros de software acabam sendo vulnerabilidades de segurança. Um *bug* deve ter algum impacto ou propriedades relevantes para a segurança para ser considerado um problema de segurança; em outras palavras, ele permite que os invasores façam algo que normalmente não seriam capazes de fazer (DOWD; MCDONALD; SCHUH, 2006).

As vulnerabilidades encontradas em um software devem ser mitigadas ou reduzidas até um nível de segurança aceitável o mais rapidamente possível. A nova versão do software com as correções implementadas deve ser disponibilizada para seus usuários substituindo a vulnerável (JÚNIOR; DOS, 2013). Entretanto, segundo Calil (2023), em geral as medidas de segurança cabidas para resolver as vulnerabilidades são tomadas de maneira reacional, ou seja, só são implementadas na ocasião em que suas vulnerabilidades são exploradas.

Para evitar e minimizar a presença de vulnerabilidades no software antes que os ataques ocorram começaram a surgir propostas de métodos de desenvolvimento de software focados em segurança. Nestes métodos a segurança não é deixada para ser pensada no final do desenvolvimento, mas permeia todo o processo, desde seu planejamento até a sua disponibilização ao usuário final. Com isso, nasce o que é conhecido como ciclo de vida do desenvolvimento de software seguro.

## 2.3 Ciclo de vida de desenvolvimento de software seguro

Os processos de ciclo de vida de software seguro são abordagens proativas para desenvolver um produto seguro, tratando o problema do software vulnerável e mal projetado na fonte, em vez de criar uma solução temporária para interromper os problemas por meio de uma abordagem reativa de penetração e correção. Esses processos trabalham profundamente a segurança do software em todo o processo de desenvolvimento do produto e incorporam pessoas e tecnologia para enfrentar e prevenir problemas de segurança de software (Laurie Williams, 2021). Um ciclo de vida de desenvolvimento do software seguro abrange a prevenção, a detecção e mitigação de defeitos de segurança do produto

Um dos principais ciclos de vida presentes hoje no mercado é o *Microsoft Security Development Lifecycle* (Microsoft, 2023c), também conhecido pela sigla SDL.

### 2.3.1 Microsoft Security Development Lifecycle (SDL)

Após a Microsoft ter algumas fraquezas de seus produtos exploradas no início dos anos 2000, principalmente o *Internet Information Services (IIS)*, seu modo de desenvolvimento mudou, seu foco passou ser, como nomeado por seu CEO, “Computação confiável” (Laurie Williams, 2021). Este novo conceito foi descrito por GATES (2002) como:

“A Computação Confiável é a prioridade mais alta para todo o trabalho que estamos fazendo. Devemos levar a indústria a um nível totalmente novo de confiabilidade em computação... Computação confiável é uma computação tão disponível, confiável e segura quanto eletricidade, serviços de água e telefonia” (GATES, 2002).

Fiel à intenção original de Gates, o Microsoft SDL forneceu a base para o setor de tecnologia da informação, oferecendo o primeiro ciclo de vida prescritivo e documentado de forma abrangente (Laurie Williams, 2021). Este modelo de desenvolvimento é composto por 12 práticas e técnicas listadas a seguir:

- **Prover Treinamento:**

A segurança é responsabilidade de todos, sejam gerentes ou desenvolvedores. Para isso, todos devem conhecer os fundamentos de desenvolvimento de um software seguro, serem capazes que enxergar a partir da perspectiva de um usuário mal intencionando e identificar seus objetivos (Microsoft, 2023c). Entretanto nem todos os profissionais envolvidos no processo de criação de um software possuem esse conhecimento, uma vez que a educação formal não inclui cibersegurança. Para que este ponto seja corrigido é necessário que sejam promovidos treinamentos de segurança de software (Laurie Williams, 2021).

- **Definir requisitos de segurança**

Os requisitos de segurança devem ser definidos durante o planejamento do projeto, quando o design inicial do sistema é escolhido. Estes são influenciados pelos requisitos funcionais específicos do sistema, os requisitos legais e de conformidade do setor, os padrões internos e externos, incidentes de segurança anteriores e ameaças conhecidas (Laurie Williams, 2021). Os requisitos de segurança devem ser atualizados à medida que o projeto vai correndo para corresponder as mudanças nos requisitos

funcionais, nos padrões adotados pelo projeto e seus possíveis cenários de ameaças (Microsoft, 2023c).

- **Definir métricas e relatórios de conformidade**

Métricas são importantes para a gestão do projeto. Assim como em outras indústrias, a de software também utiliza a estratégia de pontuar objetivos e utilizar métricas para acompanhar seu progresso. Neste contexto, existem metas de segurança e métricas para medir se estas estão sendo cumpridas ou não, definir os níveis mínimos aceitáveis de qualidade de segurança e responsabilizar as equipes de engenharia pelo cumprimento desses critérios. Definir metas e métricas no início do projeto ajuda a equipe a rastrear os defeitos durante o desenvolvimento (Microsoft, 2023c). O rastreamento de defeitos deve rotular claramente os defeitos de segurança e os itens de trabalho de segurança para permitir uma priorização assertiva, um gerenciamento de riscos efetivo e a produção de relatórios do trabalho de segurança precisos (Laurie Williams, 2021).

- **Modelagem de ameaças**

Por meio do uso de modelagem de ameaças, as equipes consideram, documentam e discutem as implicações de segurança dos projetos no contexto de seu ambiente operacional planejado e de maneira estruturada (Microsoft, 2023c). As equipes devem considerar as motivações de seus adversários e os pontos fortes e fracos dos sistemas para se defender dos cenários de ameaças associados. Alguns dos pontos observados durante essa etapa são os usuários mal intencionados, o design do sistema e seus componentes, os limites de confiança do sistema, o fluxo de dados do sistema dentro e através dos limites de confiança (JASPER, 2014).

- **Estabelecer requisitos de design**

Os requisitos de design guiam a implementação de funcionalidades de segurança e devem levar em consideração a ameaças conhecidas do sistema operacional pretendido (Laurie Williams, 2021). É importante estar atento se a implementação de um novo requisito não gerou outras fragilidades no sistema, enfatizando que a segurança é uma questão contínua e deve ser acompanhada durante todo o projeto (CALIL, 2023).

- **Definir e usar padrões de criptografia**

O uso de criptografia é um importante recurso de design de um sistema para garantir que dados confidenciais de segurança e privacidade sejam protegidos contra divulgação ou alteração não intencional quando transmitidos ou armazenados (Laurie Williams, 2021). Entretanto, é importante saber escolher o padrão adequado para o projeto, para que esta não se torne fraca e até mesmo ineficaz. Outro ponto a



ver observado na escolha da biblioteca de criptografia é se esta pode ser facilmente alterada se necessário e se segue padrões consagrados (Microsoft, 2023c).

- **Gerenciar riscos de segurança provenientes da utilização de componentes de terceiros**

A maior parte dos softwares desenvolvidos utilizam componentes de terceiros, como bibliotecas. Deste modo, é importante estar atento às dependências dos projetos, documentá-las e criar planos de mitigação de possíveis vulnerabilidades presentes, pois podem servir de porta de entrada para ataques (CALIL, 2023).

- **Utilizar ferramentas aprovadas**

Deve-se ter uma lista de ferramentas aprovadas, suas verificações e configurações de segurança associadas. O uso da versão mais recente dessas ferramentas deve ser incentivado, visando novas análises e funcionalidades (Microsoft, 2023c).

- **Executar análise estática de testes de segurança - (SAST)**

A análise estática de teste de segurança, ou SAST, é a técnica de revisão de código atenta às políticas de codificação segura, que ocorre antes da compilação. O SAST tem como objetivo automatizar a identificação de padrões de códigos inseguros, podendo ser aplicado em esteiras de integração contínua (CI) (CALIL, 2023).

Quando o SAST é integrado ao *pipeline* de *commit*, identifica as vulnerabilidades sempre que o software é criado ou empacotado (Microsoft, 2023c). Quando integrado ao ambiente de desenvolvimento detecta certas falhas como a presença de funções inseguras ou proibidas e propõe alternativas mais seguras durante o processo de codificação (Microsoft, 2023c).

- **Executar análise dinâmica de testes de segurança - (DAST)**

A análise dinâmica de teste de segurança, ou DAST, assim como o SAST é uma ferramenta de testes de segurança. Mas diferentemente do teste de segurança estático, o DAST analisa o código em tempo de execução do software compilado (CALIL, 2023). Isso ocorre para verificar as funcionalidades que só são aparentes quando todos os componentes estão integrados e em execução. O DAST é realizado com auxílio de conjuntos de ataques pré-construídos ou com ferramentas que monitoram o comportamento da aplicação em busca de corrupção de memória, problemas de privilégio do usuário e outros problemas severos de segurança (Microsoft, 2023c).

- **Executar testes de penetração**

O teste de penetração manual é uma técnica de teste caixa preta, isto é, não analisa o código diretamente. Este teste é realizado por profissionais especializados e simula

ações de um invasor com objetivo de descobrir todo e qualquer tipo de vulnerabilidade, sejam pequenas como *bugs* de implementação até grandes como falhas de design, configuração, sistema e etc (Laurie Williams, 2021). Os testes de penetração normalmente são executados junto com revisões de código automatizadas e manuais, assim fornecem uma análise melhor do que normalmente seria possível (Microsoft, 2023c). Segundo Laurie Williams (2021), o teste de penetração pode encontrar a mais ampla variedade de vulnerabilidades, embora geralmente menos eficiente em comparação com SAST e DAST.

- **Estabelecer um padrão para o processo de resposta à incidentes**

Mesmo com todas as precauções tomadas, ainda assim é possível que haja um ataque bem sucedido ao software, sendo necessário elaborar um plano de contingência. Este plano deve conter orientações de como agir quando ocorrer um incidente, contando com as informações de quem contactar nesta situação e estabelecer um padrão de mitigação da vulnerabilidade encontrada, que facilite a comunicação da equipe e a correção do *bug* (Microsoft, 2023c).

## 2.4 Modelagem de ameaças

A modelagem de ameaças é usada em ambientes que há a presença de riscos de segurança severos. Pode ser aplicada no nível do componente, do aplicativo ou do sistema (Microsoft, 2023c). Esta prática objetiva a melhoria contínua do desenvolvimento e manutenção do software visando a segurança da informação.

Segundo Shostack (2014), a modelagem de ameaças é o uso da abstração para melhor entender os riscos. Esta técnica parte do princípio que entender bem os riscos é fundamental para se proteger de maneira adequada contra as ameaças (JASPER, 2014). Isto ocorre, pois facilita a gestão de riscos, a prevenção de ataques, a aplicação das técnicas mais apropriadas para mitigar cada tipo de vulnerabilidade e a elaboração de planos de contingência que prevêm situações de ataque e descrevem como reagir nestas situações indesejadas.

Shostack (2014) lista quatro motivos para o uso da modelagem de ameaças, sendo estes, encontrar *bugs* de segurança cedo, entender seus requisitos de segurança, resolver problemas que outras técnicas não conseguem, projetar e entregar produtos melhores. Desta maneira, o uso desta técnica no início de um projeto de software ajuda na economia de recursos como tempo, equipe e dinheiro. Pois os esforços estarão voltados para riscos reais identificados daquele produto e não em técnicas genéricas de controle de segurança (OWASP, 2010).

Jasper (2014) aponta que a efetividade da técnica de modelagem de ameaças é

proveniente da revisão da arquitetura do sistema, identificação e análise de suas ameaças, a projeção de medidas de segurança e testes de segurança orientados, tudo isso pela perspectiva de um usuário mal intencionado. Deste modo, identifica os pontos mais frágeis e expostos que necessitam de maior atenção.

A modelagem de ameaças encontra todos os pontos importantes do sistema com o auxílio da elaboração de um diagrama de fluxo de dados. Com o diagrama pronto e os pontos chaves do projeto identificados, é feita uma análise de cada um destes, listando todas as possíveis ameaças associadas. O levantamento das ameaças pode ser por meio de *brainstorm*, discussões sobre os pontos chaves ou com a ajuda de ferramentas como *Microsoft Threat Modeling Tool (TMT)* e frameworks como STRIDE da *Microsoft* e o *OWASP Security Knowledge Framework*, árvores de ataques e outros. Com as possíveis ameaças elencadas, estas são priorizadas a partir do seu nível de criticidade, para isso podem ser utilizadas metodologias como a DREAD da *Microsoft*. Com as ameaças identificadas e ranqueadas são discutidas as melhores maneiras de resolvê-las, até que todas as vulnerabilidades sejam contempladas com uma possível solução ou classificadas como risco aceito e que não necessitem de nenhuma ação (SOUZA, 2023).

#### 2.4.1 Microsoft Threat Modeling Tool (TMT)

“O Microsoft TMT é uma ferramenta madura que evoluiu ao longo de vários anos” (SHI et al., 2022). Esta ferramenta funciona em um ciclo que envolve a elaboração de um diagrama, a identificação de ameaças, a busca por maneiras de mitigar as ameaças encontradas e validação de cada proposta de mitigação (Microsoft, 2023b), conforme ilustrado na Figura 2.4.1.

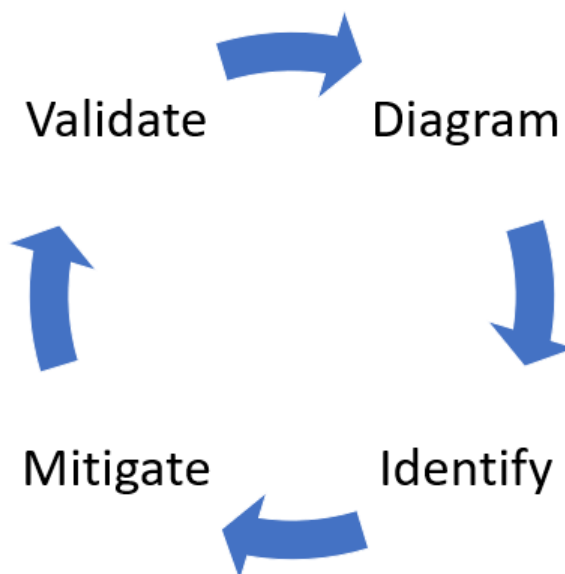


Figura 1 – Ciclo de processos do Microsoft TMT (Microsoft, 2023b)

Para começar, é construído um diagrama com base no *template* padrão disponibilizada pela ferramenta, também conhecido como modelo, ou por outro disponibilizado pela comunidade. É possível utilizar outros *templates*, pois o modelo pode ser alterado para atender a demanda do projeto e suas especificações. O TMT identificará as ameaças por meio da verificação das condições das ameaças no modelo. As ameaças encontradas são categorizadas pelo STRIDE. Assim como o modelo, os elementos do diagrama, os tipos de ameaças e as propriedades das ameaças são todos editáveis. As propriedades da ameaça incluem descrição, prioridade e contramedidas (SHI et al., 2022).

## 2.4.2 STRIDE

A abordagem STRIDE foi desenvolvido pela *Microsoft* e visa categorizar os diferentes tipos de ameaças e simplificar a segurança geral. STRIDE é um acrônimo para as seguintes seis ameaças (XIN; XIAOFANG, 2014):

- ***Spoofing* (Falsificação):** O uso ilegal de dados de autenticação para acessar o sistema.
- ***Tampering* (Adulteração):** É a alteração de dados ou do sistema de maneira maliciosa.
- ***Repudiation* (Repúdio):** Negação plausível de ações tomadas sob um determinado usuário ou processo.
- ***Information disclosure* (Divulgação de Informações Confidenciais):** Vazamento de dados, isto é, as informações são acessadas por usuários que não possuem autorização.
- ***Denial of Service* (Negação de Serviço):** Os usuários perdem acesso ao serviço.
- ***Elevation of Privilege* (Elevação de Privilégio):** Um usuário ou processo passa a ter privilégios adicionais que não deveria, de modo que passa a ter capacidade suficiente para danificar ou apagar todo o sistema.

Segundo Hewko (2021), cada ameaça do STRIDE está vinculada à quebra de um pilar do sistema, como está retratado na Tabela 1.

Tabela 1 – Pilares de segurança e ameaças, segundo o framework STRIDE (HEWKO, 2021)

Propriedade	Ameaça	Definição
Autenticação	Falsificação	Personificar algo ou alguém
Integridade	Adulteração	Modificar dados ou código
Não repúdio	Repúdio	Alegar não ter realizado uma ação
Confidencialidade	Divulgação de Informações	Expor informações a alguém não autorizado a vê-las
Disponibilidade	Negação de Serviço	Negar ou degradar o serviço aos usuários
Autorização	Elevação de Privilégio	Obter recursos sem a autorização adequada

### 2.4.3 DREAD

O modelo de risco DREAD é o utilizado pela *Microsoft*, este é usado para ranquear as ameaças encontradas. O acrônimo DREAD significa *Damage potential* (Dano potencial), *Reproducibility* (Reproduzibilidade), *Explorability* (Explorabilidade), *Affect user* (Usuários afetados) e *Discoverability* (Descoberta) (RANSOME et al., 2014). Para que as ameaças sejam ranqueadas é necessário responder uma pergunta para cada letra do acrônimo, a resposta deve ser um número de 0-10, quanto maior o número mais sério é o risco. Seguindo a ordem do acrônimo as perguntas são: “Se ocorrer a exploração da ameaça, quanto dano isso causará?”, “Quão fácil é reproduzir a exploração desta ameaça?”, “O que é necessário para explorar esta vulnerabilidade?”, “Quanto usuários serão afetados?” e “Quão fácil é descobrir esta ameaça?” (RANSOME et al., 2014).

O cálculo do valor geral do risco da ameaça é realizado a partir da seguinte fórmula:

$$\text{Risco-DREAD} = (\text{Dano potencial} + \text{Reproduzibilidade} + \text{Explorabilidade} + \text{Usuários afetados} + \text{Descoberta})/5$$

O próximo passo é classificar as ameaças em nível de risco, podendo ser baixo (valor = 1), médio (valor =2) ou alto (valor=3). Para facilitar o cálculo do risco total é recomendado utilizar a tabela 2.

Tabela 2 – Tabela para cálculo do nível do risco

Categoria DREAD	Baixo risco (1)	Médio risco (2)	Alto Risco (3)	Risco Subtotal
Dano potencial				
Reproduzibilidade				
Explorabilidade				
Usuários afetados				
Descoberta				
			<b>Risco total:</b>	

Para cada ameaça será preenchida uma matriz como demonstrado na tabela 2, marcando qual o nível do risco para cada categoria. Se o risco for baixo valerá 1, se médio valerá 2 e se alto valerá 3. Com o nível de risco das cinco categorias identificados, a pontuação subtotal será somada resultando no risco total, esta pontuação varia de 5 á 15. Se a pontuação total for de 12 á 15 é considerada alta, se de 8 á 11 é média e de 5 á 7 é baixa.

## 2.5 Testes de segurança estático

*Static Analysis Security Testing*, mais conhecida como SAST ou testes de segurança estático é a análise da aplicação antes da sua execução, capaz de detectar pontos de atenção no código. Estes pontos, se não trabalhados, podem se tornar vulnerabilidades (RIBEIRO, 2015).

Segundo Ribeiro (2015), existem diferentes formas de analisar estaticamente um código, variando de uma simples análise léxica ou detecção de padrões, análises de fluxo de dados à análise de grafos de fluxo de controle. Entretanto, nenhuma técnica é capaz de encontrar 100% das fraquezas presentes num código fonte, por isso é recomendado o uso de diferentes técnicas (RIBEIRO, 2015).

O SAST pode ser realizado de maneira manual a depender do contexto que está inserido, como a inspeção de um determinado trecho de código, mas normalmente é automatizada por meio de ferramentas (BARBOSA, 2015). Estas ferramentas são divididas em várias categorias, cada uma com suas características e intuito. Segundo Barbosa (2015), algumas dessa são:

- **Checagem de tipo:** certifica os tipos das variáveis, desde suas declarações até suas atribuições e conversões, quando existem no código.
- **Checagem de estilo:** observa espaços em branco, nomenclaturas, funções depreciadas, estrutura do programa e outros. Não compromete a execução do programa, mas podem dificultar a legibilidade do código e sua manutenção.
- **Entendimento do programa:** Esta categoria busca associar o código fonte com o comportamento do programa em alto nível.
- **Verificação do programa:** checa se o código do programa atende às especificações de implementação e comportamento previamente estabelecidos.
- **Checagem de propriedades:** analisa o código a partir de especificações parciais de suas características, detalhando assim apenas parcialmente o comportamento do software.

- **Descoberta de erros:** busca encontrar *bugs* no código que foram introduzidos de maneira acidental ao ser implementado erroneamente.
- **Revisão de segurança:** Utiliza-se de técnicas presentes nas outras categorias de ferramentas, mas com objetivo de identificar falhas de segurança.

### 2.5.1 SonarQube

O *SonarQube Community Edition* <sup>1</sup> é uma ferramenta gratuita e de código aberto, desenvolvida pelo *Sonar Source*. Segundo a própria ferramenta, o objetivo é fornecer às equipes de desenvolvimento uma solução integrada ao fluxo de trabalho de modo que o código seja limpo e de qualidade. O Sonar possui suporte às principais linguagens de programação, estruturas e tecnologias de nuvem.

Esta ferramenta realiza a análise em tempo de execução e observa se há duplicidade de código, se algum trecho do código fonte possa gerar *bugs* ou falhas de segurança, por exemplo. Para que esta análise seja realizada é necessário configurar as métricas de qualidade, que podem seguir padrões pré-definidos ou customizar para o software em questão.

## 2.6 Testes de segurança dinâmicos

*Dynamic Analysis Security Testing*, mais conhecido como DAST ou teste de segurança dinâmico é a análise da aplicação durante sua execução. Para que esta técnica possa ser utilizada é necessário que o código possa ser executado e instalado (RIBEIRO, 2015).

O DAST tem como objetivo encontrar as fraquezas da aplicação, observando a partir da perspectiva de um invasor (SONI, 2021), testar os mecanismos de defesa, verificar planos de resposta e confirmar a adesão às políticas de segurança (OWASP, 2023).

O teste dinâmico é realizado com a aplicação de ataques mal intencionados ao software e compara os resultados obtidos destes ataques com os esperados. Se os resultados forem diferentes significa que há pontos de atenção no projeto, onde um intruso pode atacar, roubar dados, obter acesso e negar o serviço para os usuários (SONI, 2021). Estes ataques são automatizados e realizados pelas ferramentas DAST, não requerendo nenhuma informação sobre o código-fonte ou codificação interna do sistema como é o caso dos testes SAST (SONI, 2021).

Segundo OWASP (2023), normalmente o teste dinâmico segue as etapas:

<sup>1</sup> <https://www.sonarsource.com/open-source-editions/sonarqube-community-edition/>

- Explorar – O testador estuda o sistema que será testado. Isto é, identifica os *end-points*, os *patches* estão instalados, etc. Além disso, o testador procura no site por conteúdos ocultos, lista as vulnerabilidades conhecidas e outras indicações de fraqueza.
- Ataque – Nesta fase são realizados ataques mal intencionados ao software, que busca explorar as vulnerabilidades conhecidas ou suspeitas para provar que elas existem.
- Relatório – Os resultados dos testes são relatados, incluindo as vulnerabilidades, como as exploraram e quão difíceis foram as explorações, e a gravidade da exploração.

Um ponto de atenção deste tipo de teste é que apenas os *bugs* que estão naquele determinado fluxo de execução das entradas fornecidas pelos testes podem ser identificadas (RIBEIRO, 2015). Isto é, os testes normalmente não conseguem abranger todas as entradas possíveis e desta maneira pode gerar falsos negativos, quando o defeito não é identificado pela ferramenta (BARBOSA, 2015). Além disso, há a dificuldade de replicar a análise em diferentes contextos e ambientes, uma vez que a análise depende diretamente da execução do programa em questão (BARBOSA, 2015).

### 2.6.1 OWASP Zap

O *OWASP Zed Attack Proxy*<sup>2</sup>, ZAP, é uma ferramenta gratuita de DAST destinada para testar aplicações web. Esta ferramenta funciona como “*proxy man-in-the-middle*”, isto é, fica entre o navegador do testador e o aplicativo da web, como demonstra a figura 2.6.1. Quando a aplicação possui *proxy* o ZAP pode ser conectada a este assumindo a posição entre o navegador do testador e o *proxy*, como ilustra a figura 2.6.1. Desta maneira, pode interceptar, inspecionar e alterar as mensagens enviadas entre o navegador e o aplicativo da web e então encaminhar esses pacotes para o destino, se assim desejar (OWASP, 2023).

O OWASP ZAP pode ser usado como um aplicativo independente ou como um processo paralelo a aplicação.



Figura 2 – Posição do ZAP (OWASP, 2023)

<sup>2</sup> <https://www.zaproxy.org/>





Figura 3 – Posição do Zap quando há *proxy* (OWASP, 2023)

Esta ferramenta pode ser utilizados em diferentes sistemas operacionais. Além disso, possui complementos gratuitos e por ser de código aberto é possível contribuir com novas funcionalidades para atender as demandas da aplicação web testada (OWASP, 2023).

## 3 Metodologia

O presente capítulo tem como objetivo descrever a metodologia que será utilizada para o desenvolvimento do estudo proposto. Para isso, a metodologia escolhida foi o estudo de caso.

A metodologia de estudo de caso é amplamente utilizada em diferentes áreas, tais como, política, sociologia, psicologia e também engenharia de software. O estudo de caso é um método empírico que visa investigar os acontecimentos em seu próprio contexto (WOHLIN et al., 2012). Isso significa, que é um método que possui a vida real como contexto, o que resulta em um design flexível, uma vez que possui caráter observacional e cada caso contém suas particularidades.

### 3.1 Protocolo de Brereton

Para realizar o presente estudo será utilizado o protocolo de Brereton et al. (2008). Este protocolo é composto dos seguintes 11 passos:

1. **Base:** Este passo visa entender o contexto que o estudo está inserido, definir as questões que guiaram a pesquisa e que devem ser respondidas por esta. Ou seja, este tópico é subdividido nos seguintes três itens:
  - Buscar por pesquisas anteriores sobre o tema.
  - Definir a principal questão a ser respondida pelo estudo.
  - Identificar questões adicionais.
2. **Design:** Esta fase tem como objetivo definir as características do estudo de caso que será realizado e para isso deve realizar três atividades:
  - Estabelecer qual será o tipo do estudo de caso, se único ou múltiplo, se será explanatório, descritivo, exploratório ou casual.
  - Descrever o objeto de estudo.
  - Identificar as proposições geradas pelas perguntas norteadoras do estudo e especificar as medidas a serem usadas para investigar estas proposições.
3. **Escolha do caso:** Nesta fase é feito o processo seletivo do caso ou os casos que serão objeto de estudo. Para isso é necessário definir os critérios que estes casos devem atender.

4. **Procedimentos e funções do estudo de caso:** Consiste em definir os procedimentos que regem os procedimentos de campo e designar os papéis dos membros da equipe de pesquisa do estudo de caso.
5. **Coleta de dados:** Execução da coleta de dados, tanto qualitativos quanto quantitativos. Esta fase pode ser dividida em três sub fases:
  - Identificar os dados a serem coletados.
  - Definir o plano de coleta de dados.
  - Estabelecer como os dados deverão ser armazenados.
6. **Análise dos dados:** São realizadas tanto análise qualitativas quanto quantitativas. [Brereton et al. \(2008\)](#) divide esta etapa em três sub etapas:
  - Elencar os critérios de interpretação dos resultados obtidos.
  - Selecionar quais elementos de dados são usados para abordar qual proposição de pesquisa e como estes dados serão combinados para responder à pergunta
  - Considerar a gama de resultados possíveis e buscar por explicações alternativas para os resultados e identificar qualquer informação necessária para distinguir eles.

A análise deve ocorrer à medida que a tarefa do estudo de caso progride.

7. **Validade do plano:** Este tópico pode ser subdividido em quatro validações diferentes:
  - **Validação geral:** Aqui é feita uma revisão do design e do plano de coleta de dados para verificar que estes atendem aos objetivos do estudo e não estão deixando nenhum detalhe passar batido.
  - **Validação da construção:** Validar a escolha das medidas operacionais planejadas para o estudo de caso por meio de pesquisas bibliográficas sobre tais.
  - **Validação interna:** Esta validação é para apenas estudos de carácter explicativo ou casual. E tem como objetivo mostrar uma relação causal entre resultados e a intervenção.
  - **Validação externa:** Identifica o domínio em qual o estudo de caso se encaixa. Para isso, são usadas as táticas do uso da teoria para estudos de caso único e o uso de estudos de casos múltiplos para investigar resultados em diferentes contextos.
8. **Limitações do Estudo:** Esta fase especifica as questões de validade residual, isto é, potenciais conflitos de interesse, sejam inerentes ao problema ou decorrentes do plano.

9. **Reportar:** Identifica qual é o público-alvo e sua relação com estudos renomados.
10. **Agendar:** Este passo estabelece um cronograma a partir das atividades essenciais para o estudo de caso, como o seu planejamento, a coleta de dados, a análise de dados e o reporte das informações obtidas.
11. **Apêndice:** Este tópico é dividido em dois subitens:
  - **Validação:** Onde são relatados os resultados do plano de verificação.
  - **Divergências:** este item é atualizado durante todo o estudo. Isso ocorre, pois a qualquer momento pode ser encontrada uma diferença entre o esperado e o obtido nas fases acima, quando estas divergências são encontradas elas são documentadas neste apêndice.

## 3.2 Aplicação do protocolo ao estudo

Neste tópico será detalhado o uso do protocolo de [Brereton et al. \(2008\)](#) na avaliação da segurança do *VALIDAR - Serviço de validação de assinaturas eletrônicas* provido pelo Instituto Nacional de Tecnologia da Informação.

A permissão para divulgação de informações sobre a segurança do Validar foi obtida junto ao responsável pelo serviço no Instituto Nacional da Informação (ITI). O Termo de Permissão para Divulgação de Informações sobre Vulnerabilidades do Serviço VALIDAR se encontra no Anexo A.

1. **Base:** Para a realização deste tópico foram utilizadas pesquisas bibliográficas sobre a temática por meio de bases bibliográficas como Google Acadêmico, periódicos CAPES e outros. As informações relevantes e condizentes com o tema foram compiladas no capítulo 2, referencial teórico.

Como principal questão a ser respondida pelo presente trabalho é “O VALIDAR, Serviço de validação de assinaturas eletrônicas, possui falhas de segurança severas?”. Além disso, o estudo também deverá responder as seguintes questões caso a resposta para a pergunta acima seja sim, “Como essas falhas impactam o resultado do serviço prestado?”, “Quais são os principais fatores que contribuem para estas vulnerabilidades?”, “É possível melhorar o nível de segurança do serviço?” e “Quais ações são necessárias para torná-lo mais seguro?”.

2. **Design:** O presente estudo de caso será do tipo simples, isto é, composto por apenas um caso. Sendo de carácter descritivo, ou seja, descreve sequência de eventos e mecanismos subjacentes. A descrição do objeto de estudo é encontrada no capítulo 4, Análise do VALIDAR.

3. **Escolha do caso:** A escolha do caso foi realizada por conveniência, uma vez que, a autora possui acesso aos códigos fontes do serviço.
4. **Procedimentos e funções do estudo de caso:** Os procedimentos aplicados serão os seguintes:
  - Aplicação da modelagem de ameaças por meio da ferramenta *Microsoft Thread Modeling Tool* em conjunto com o *framework* STRIDE também da *Microsoft* para levantamento e priorização de pontos de atenção;
  - Realização de testes estáticos de segurança, SAST, com o *Sonar Source*;
  - Documentar os resultados do SAST;
  - Realização de testes dinâmicos de segurança, DAST, por meio do OWASP Zap;
  - Registrar os resultados do DAST;
  - Analisar os resultados individuais dos testes;
  - Analisar os resultados dos testes em conjunto;
  - Identificar os principais pontos de atenção;
  - Apresentar propostas para melhoria do nível de segurança da aplicação.
5. **Coleta de dados:** Os dados a serem coletados são:
  - Vulnerabilidades;
  - Quantidade de vulnerabilidades;
  - Tipos de vulnerabilidades;
  - Nível de gravidade da Vulnerabilidade.

Esses dados serão coletados com as ferramentas *Microsoft Thread Modeling Tool* e o *framework* STRIDE, OWASP ZAP e *Sonar Source*. Para armazená-los serão utilizados tabelas e gráficos.

6. **Análise de dados:** Esta etapa buscará identificar se há predominância de um ou mais tipos de pontos de atenção e o quão graves são. Além disso, observará se existe alguma relação, seja direta ou indireta, entre os tipos e os níveis de gravidade.
7. **Validade do plano:** Os dados a serem coletados deverão ser suficientes para responder a questão principal, “O VALIDAR, Serviço de validação de assinaturas eletrônicas, possui falhas de segurança severas?” e também as perguntas complementares, especificando quais pontos de atenção estão levando a tal resultado. Com esta avaliação de segurança também será respondido se é possível tornar o serviço mais seguro e quais ações serão necessárias para isto.

8. **Limitações do Estudo:** O estudo só deverá ser publicado após as correções das vulnerabilidades encontradas. Para que assim os cibercriminosos não explorem estas brechas de segurança e lesem cidadãos de bem ou até mesmo o serviço público.
9. **Reportar:** O público-alvo deste estudo de caso são as equipes de desenvolvimento do serviço, tanto por parte do ITI (Instituto de Tecnologia da Informação), quanto pelo LabSEC ( Laboratório de Segurança em Computação) e todos os usuários do serviço VALIDAR, sejam pessoas físicas ou jurídicas.

O ITI autorizou a divulgação do nome do serviço e das possíveis vulnerabilidades encontradas, conforme demonstra o anexo A - Primeiro Anexo (9), de título: Termo de Permissão para Divulgação de Informações sobre Vulnerabilidades do Serviço VALIDAR.

## 4 Análise do VALIDAR

O VALIDAR, serviço de validação de assinaturas digitais, foi desenvolvido pelo Instituto Nacional de Tecnologia da Informação (ITI) em parceria com o Laboratório de Segurança da Computação (LabSEC) da Universidade de Santa Catarina.

O ITI é uma autarquia federal, vinculada à Casa Civil da Presidência da República, responsável por manter e executar as políticas da Infraestrutura de Chaves Públicas Brasileira – ICP-Brasil. O instituto também é a autoridade certificadora (AC) raiz de cadeia de certificação digital, isto é, a primeira autoridade da cadeia de certificação (ITI, b).

Segundo o Decreto 10543/20<sup>1</sup> Art 9º. item II, O ITI poderá atuar, em conformidade com as políticas e as diretrizes do Governo federal, junto a pessoas jurídicas de direito público interno no apoio técnico e operacional relacionado à criptografia, à assinatura eletrônica, à identificação eletrônica e às tecnologias correlatas.

O LabSEC faz parte do Departamento de Informática e de Estatística (INE) da Universidade de Santa Catarina (UFSC). O seu objetivo é estudar, pesquisar, avaliar e desenvolver soluções focadas em segurança da computação. Suas principais atuações são criptografia, assinatura digital segurança em sistemas computacionais, infraestrutura de chaves públicas (ICPs) e protocolos criptográficos <sup>2</sup>.

O VALIDAR é a junção de dois serviços providos pelo ITI, o Verificador de Conformidade e o Validador de Documentos Digitais de Saúde, conforme ilustra a Figura 4. O verificador de conformidade, representado pelo círculo branco, é o núcleo da API do VALIDAR responsável por verificar se os documentos assinados eletronicamente respeitam o Padrão Brasileiro de Assinatura Digital (PBAD) e a Infraestrutura de Chaves Públicas Brasileira (ICP-Brasil). Enquanto o Validador de documentos digitais de saúde é uma segunda camada da API do VALIDAR, ilustrado pelo círculo azul da Figura 4, esta camada valida se as assinaturas eletrônicas presentes nos documentos pertencem à um profissional de saúde habilitado.

<sup>1</sup> <[https://www.planalto.gov.br/ccivil\\_03/\\_ato2019-2022/2020/decreto/D10543.htm](https://www.planalto.gov.br/ccivil_03/_ato2019-2022/2020/decreto/D10543.htm)>

<sup>2</sup> <https://labsec.ufsc.br/>



Figura 4 – Representação da API do VALIDAR

## 4.1 Assinaturas eletrônicas

As assinaturas eletrônicas são baseadas em dois pilares, a autoria e a integridade. O pilar da autoria é responsável por conferir condição de autor à pessoa que assina, de modo que comprove que a assinatura pertence ao signatário. Isto é possível por meio de um par de chaves, composto por uma chave pública e outra privada, que são complementares uma à outra (BEHRENS, 2005). O pilar da integridade assegura que o documento está íntegro, isto é, seu conteúdo não foi alterado após a assinatura. Este pilar é garantido pelo cálculo da HASH do documento. Deste modo, quando há alguma alteração no conteúdo do documento, a integridade é comprometida e quando há adulteração da assinatura a autoria é corrompida. Uma vez que se a integridade foi alterada a assinatura já não pertence ao documento. Em ambos os casos o VALIDAR evidencia que algo não está como deveria.

Estes dois pilares ficam evidentes na própria lei que criou a Infraestrutura de Chaves Brasileira (ICP-Brasil), na MEDIDA PROVISÓRIA No 2.200-2, DE 24 DE AGOSTO DE 2001<sup>3</sup>. No Art.10, parágrafo 1o., tem-se:

Art. 10. Consideram-se documentos públicos ou particulares, para todos os fins legais, os documentos eletrônicos de que trata esta Medida Provisória. § 1o As declarações constantes dos documentos em forma eletrônica produzidos com a utilização de processo de certificação disponibilizado pela ICP-Brasil presumem-se verdadeiros em relação aos signatários, na forma do art. 131 da Lei no 3.071, de 1o de janeiro de 1916 - Código Civil.

<sup>3</sup> <[https://www.planalto.gov.br/ccivil\\_03/mpv/antigas\\_2001/2200-2.htm](https://www.planalto.gov.br/ccivil_03/mpv/antigas_2001/2200-2.htm)>



### 4.1.1 Validade jurídica

A MP 2.200-2 também assegura que as assinaturas eletrônicas ICP-Brasil possuem validade jurídica. Esta validade jurídica é pautada em cinco pilares, como demonstrado na Figura 5.

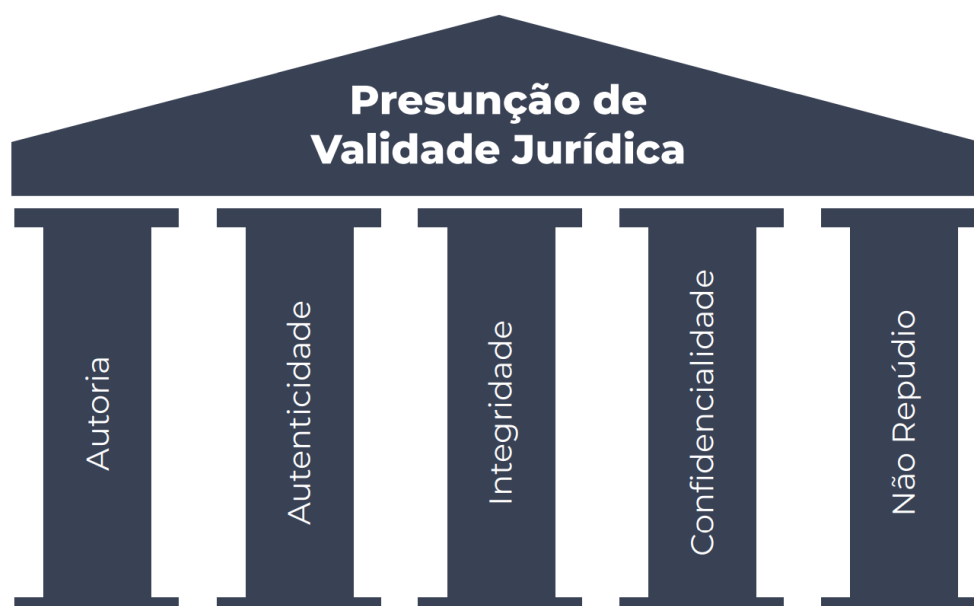


Figura 5 – Pilares da Validade Jurídica (ITI, a)

- Autoria: condição de autor.
- Autenticidade: é a certeza de que a assinatura pertence ao autor e que a assinatura refere-se ao documento.
- Integridade: é a garantia que o conteúdo do documento não foi alterado após a assinatura.
- Confidencialidade: apenas pessoas e sistemas autorizados podem acessar as informações e dados do objeto em questão.
- Não repúdio: é o princípio que impede o questionamento sobre a autoria de uma declaração, não permite o contestar da origem da assinatura.

### 4.1.2 Classificação das assinaturas eletrônicas

Assinatura eletrônica é o conjunto de dados eletrônicos que permitem a comprovação da autoria (ITI, 2021). A lei Nº 14.063<sup>4</sup> no Art. 4º estabelece três classes de assinaturas para os casos de integração com os entes de governo. As assinaturas eletrônicas podem ser classificadas como:

<sup>4</sup> <[https://www.planalto.gov.br/ccivil\\_03/\\_ato2019-2022/2020/lei/114063.htm](https://www.planalto.gov.br/ccivil_03/_ato2019-2022/2020/lei/114063.htm)>

- **Assinatura eletrônica simples** permite identificar quem está assinando e anexa ou associa seus dados a outros dados em formato eletrônico.
- **Assinatura eletrônica avançada** utiliza certificados não emitidos pela ICP-Brasil ou outro meio de comprovação da autoria e da integridade de documentos em forma eletrônica, desde que admitido pelas partes como válido ou aceito pela pessoa a quem for oposto o documento. É o caso da assinatura gov.br, “todo cidadão tem acesso a essa modalidade de assinatura desde que se qualifique no modelo de autenticação existente no portal gov.br, de modo que quando a qualificação é prata ou ouro” (ITI, 2022).
- **Assinatura eletrônica qualificada** é gerada a partir do certificado digital ICP-Brasil, nos termos do § 1º do art. 10 da Medida Provisória nº 2.200-2, de 24 de agosto de 2001. Esta classe de assinatura possui equivalência funcional à uma assinatura manuscrita (ITI, 2022).

Segundo o tópico 4 do DOC-ICP-15 (ITI, 2021),

“4.1 Assinatura Digital ICP-Brasil é a assinatura eletrônica que:

- a esteja associada inequivocamente a um par de chaves criptográficas que permita identificar o signatário;
- b seja produzida por dispositivo seguro de criação de assinatura;
- c esteja vinculada ao documento eletrônico a que diz respeito, de tal modo que qualquer alteração subsequente neste seja plenamente detectável; e
- d esteja baseada em um certificado ICP-Brasil, válido à época da sua aposição.”

O ITI (2022) também considera as assinaturas eletrônicas qualificadas de infraestruturas de chaves-públicas oficiais de outros países como assinaturas qualificadas, desde que o Brasil mantenha algum acordo de reconhecimento de assinaturas eletrônicas ou digitais, como os países do MERCOSUL.

## 4.2 Aspectos técnicos

Nesta seção serão apresentados os aspectos técnicos da serviço VALIDAR. Entre este estão as tecnologias utilizadas, a arquitetura de lançamento do serviço e a atual.

### 4.2.1 Arquitetura

O VALIDAR foi lançado com uma arquitetura simples, formada apenas pelo *frontend* com o *proxy* e a API, representado pela Figura 6. Nesta arquitetura a requisição de

validação de assinaturas era feita diretamente do *front* para a API, quando o signatário era um médico, dentista ou farmacêutico esta informação era validada pela API. A API devolvia o JSON com a resposta da validação direto para o *front*, onde era realizado alguns tratamentos e apresentado ao usuário.



Figura 6 – Representação da antiga arquitetura do VALIDAR

A partir da atualização realizada dia 18/09/2023 a arquitetura do VALIDAR foi alterada e passou a utilizar uma aplicação intermediária para realizar as requisições à API e também a ferramenta APISIX para realizar o controle de requisições. Deste modo busca aumentar a segurança no serviço, pois impede o acesso direto aos arquivos JS onde são realizados alguns tratamentos. A arquitetura atual é composta por um *front-end* e *front* juntos, o *middleware*, a APISIX e a API (ilustrada pela Figura 4). Esta arquitetura é representada na Figura 7.



Figura 7 – Representação da arquitetura atual do VALIDAR

Com este novo formato arquitetural o fluxo de requisição ficou da seguinte maneira:

1. O *front-end* recebe o documento enviado pelo usuário e repassa ao *proxy*;
2. O *proxy* faz a requisição de validação ao *middleware*;
3. O *middleware* recebe a requisição do *proxy* e a repassa à API, juntamente com sua *API key*;
4. A *APISIX* verifica se a *API key* recebida está cadastrada e se sim libera para que a requisição chegue até a API;
5. A API realiza a verificação de conformidade e validação das assinaturas eletrônicas encontradas no documento;

6. A API devolve para a *APISIX* um JSON com todas as informações sobre as assinaturas encontradas;
7. A *APISIX* repassa o JSON recebido ao *middleware*;
8. O *middleware* recebe o JSON de resposta, realiza o tratamento deste e devolve um JSON simplificado ao *front* por meio *proxy*;
9. O *front* recebe as informações do JSON tratado e apresenta ao usuário, seguindo o Padrão Digital do Governo.

### 4.2.2 Tecnologias

O VALIDAR é uma aplicação web e como tal utiliza muitas tecnologias, são essas:

- *HTML*, *CSS* e *Java Script* para implementar o *front-end*;
- Apache Server para subir o servidor do *front-end* e montar o *proxy*;
- Java Script com o *framework* node para implementar o *middleware*;
- Java e Maven para a API;
- Docker para gerar imagens e permitir que a aplicações rode independente de um sistema operacional específico;
- Git para versionamento de código;
- *Secure Sockets Layer (SSL)* para criptografar as informações que transitam no site;
- *Api gateway* para gerenciar a API, isto é, monitorar a distribuição e uso da mesma;
- *Microsoft planner* para gerenciamento do projeto
- *WhatsApp*, *Telegram*, *email* para comunicação escrita;
- *Microsoft Teams*, *Cisco Webex* e *Google Meet* para videochamadas;
- Miro para *brainstorm*;
- *Microsoft OneNote* para elaboração e armazenamento de documentos.

### 4.3 O processo de validação

A Cartilha de uso do VALIDAR (ITI, 2022) no tópico 1.5 “O que é o serviço de validação de assinaturas eletrônicas?” apresenta de forma simples o processo de validação das assinaturas eletrônicas. Este processo consiste em verificar se os requisitos técnicos estabelecidos no DOC-ICP-15 (ITI, 2021) são atendidos e estão em conformidade com os padrões nacionais e internacionais.

Os requisitos analisados durante o processo de validação, segundo ITI (2022) são:

a **Validar o estado criptográfico da assinatura.**

O DOC-ICP-15 apresenta o seguinte fluxo simplificado para a verificação de uma assinatura digital:

1. o documento digital e a assinatura eletrônica associada e o certificado digital do signatário são disponibilizados para o VALIDAR;
2. é calculado o resumo criptográfico do documento;
3. a assinatura é decifrada com a chave pública do signatário, contida no certificado digital. Assim é obtido o resumo criptográfico gerado e cifrado pelo signatário no momento da assinatura;
4. os resumos criptográficos obtidos nos passos 2 e 3 são comparados. Quando iguais, significa que o documento está íntegro e que sua autoria foi identificada por meio do certificado digital. Caso contrário, a assinatura é considerada inválida.

Este fluxo simplificado está representado na Figura 8.

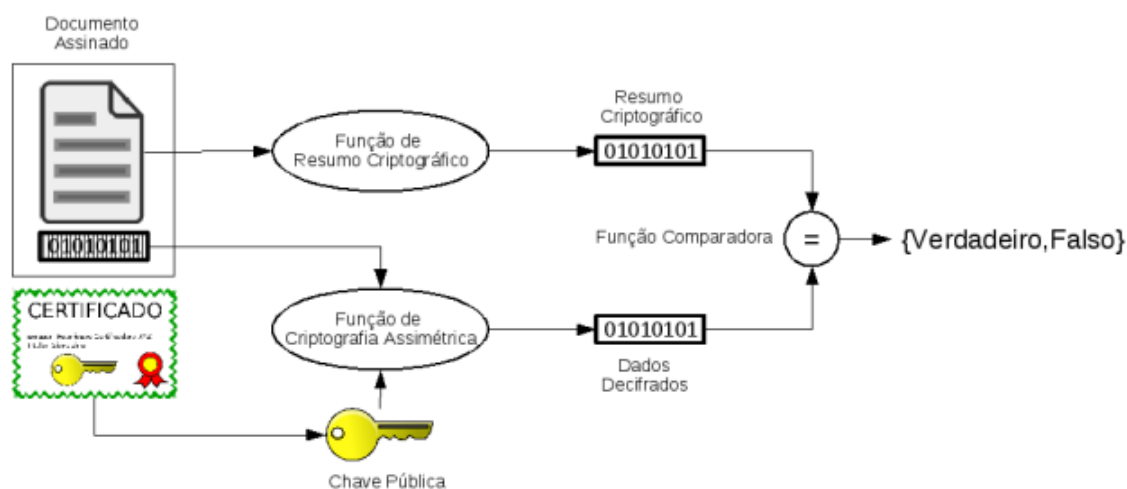


Figura 8 – Diagrama simplificado de verificação de assinatura digital (ITI, 2021)

**b Verificar a cadeia de confiança da certificação.**

Segundo ITI (2021):

“Cadeia de certificação é uma série hierárquica de certificados assinados por sucessivas Autoridades Certificadoras - ACs. A cadeia de certificação compreende o certificado da entidade final, assinado por uma AC, e zero ou mais certificados de ACs assinados por outras ACs, até o certificado de confiança, porém não incluindo este, [...]”

Este requisito é responsável por verificar se a cadeia de confiança de certificação está válida na referência temporal utilizada no momento em que a assinatura foi criada. A referência temporal pode ser o horário do computador, do serviço de criação de assinaturas ou o carimbo de tempo.

**c Confirmar a validade da assinatura eletrônica.**

Para confirmar a validade da assinatura eletrônica da Autoridade Certificadora monta-se a cadeia de certificação a partir do certificado digital do titular até chegar a AC raiz de uma determinada âncora de confiança. As âncoras de confiança reconhecidas pelo VALIDAR são: ICP-Brasil, gov.BR, países do Mercosul, e outros países que mantêm acordos de reconhecimento de assinaturas eletrônicas com o Brasil.

Com a cadeia montada é possível verificar se o certificado digital do signatário estava válido no momento da assinatura. Isto é, nenhum dos certificados da cadeia de confiança se encontra vencido ou na lista de certificados revogados.

**d Atender ao Padrão Brasileiro de Assinatura Digital (PBAD) da ICP-Brasil.**

Este requisito verifica se a assinatura eletrônica atende ao PBAD da ICP-Brasil. Isto é, se encaixar em algum dos tipos de assinaturas definidos pelo DOC-ICP-15, respeitar as políticas de assinatura e o formato de codificação do arquivo de assinatura também estabelecido no DOC-ICP-15.

## 5 Resultados da Modelagem de Ameaças

Neste capítulo será descrito o passo-a-passo realizado para a coleta de dados da modelagem de ameaças deste estudo de caso usando o *Microsoft Threat Modeling Tool* em conjuntos com os *frameworks* da *Microsoft* STRIDE e DREAD. Além disso, esses dados coletados serão registrados, destrinchados e analisados nesta seção. Para isso, foram utilizados tabelas, gráficos e o diagrama da arquitetura do sistema VALIDAR.

### 5.1 Passo-a-passo da coleta de dados

A modelagem de dados foi realizada com o auxílio da ferramenta gratuita *Microsoft Threat Modeling Tool*. O *download* do TMT pode ser realizado no próprio site da *Microsoft*, no artigo Introdução ao *Threat Modeling Tool* ([Microsoft, 2023b](#)) é possível encontrar o link para *download* da ferramenta. Neste artigo são apresentadas orientações de uso do TMT de forma detalhada e geral.

Nesta seção será descrito o passo-a-passo realizado para a modelagem de ameaças deste estudo, do serviço da validação de assinaturas eletrônicas - VALIDAR. O primeiro passo é escolher entre as opções de usar um modelo de diagrama ou criar o seu próprio modelo. Para este estudo o diagrama foi elaborado do zero, sem nenhum modelo preestabelecido.

O segundo passo é desenhar o diagrama, para isso a ferramenta oferece uma série de componentes, que são divididos em seis categorias. As categorias são: processos genéricos, interação externa genérica, armazenamento de dados genéricos, fluxo de dados genérico, limite genérico da linha de confiança, limite de fronteira de confiança genérica.

Para elaborar o diagrama que representa o fluxo de dados do VALIDAR foram utilizados os componentes, *Human User* da categoria interação externa genérica para retratar o usuário, o *CorpNet Trust Border Boundary* da categoria limite de fronteira de confiança genérica simbolizando o servidor do serviço, os componentes seguintes pertencem a categoria processos genéricos, o componente *Web Server* representa o *Proxy Apache*, o *Web Application* simboliza o *front-end* do serviço, o *Managed Application* retrata o *middleware*, a APISIX e a API e para representar as APIs externas foram utilizados componentes *External Web Application*. As interações de elementos externos com o servidor são realizadas por meio do protocolo HTTPS, enquanto as interações dentro do servidor entre os componentes são feitas por meio do protocolo HPPT. O diagrama desenhado pode ser visto na Figura 9.

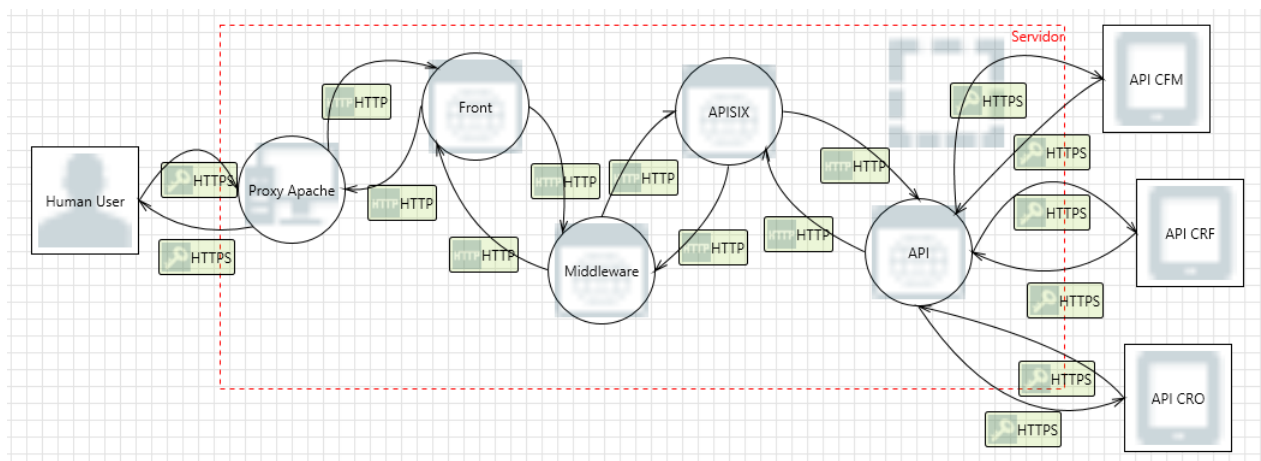


Figura 9 – Diagrama simplificado da arquitetura do sistema VALIDAR.

O terceiro passo é realizar a análise do diagrama, para isto basta clicar no ícone do arquivo com a lupa, o TMT retornará uma lista com as ameaças encontradas no modelo desenhado. As ameaças são classificadas através da abordagem do SDL chamada STRIDE (Falsificação, Adulteração, Repúdio, Divulgação de Informações, Negação de Serviço e Elevação de Privilégios). Segundo [Microsoft \(2023b\)](#):

“A ideia é que o software incorra em um conjunto previsível de ameaças, que pode ser encontrado usando estas 6 categorias.”

Ao clicar em um item devolvido pela lista de ameaças encontradas é mostrado a interação entre os dois componentes associado a ameaça selecionada e a janela “Propriedades da Ameaça” é aberta, trazendo mais informações sobre a ameaça, como exemplificado na Figura 10.

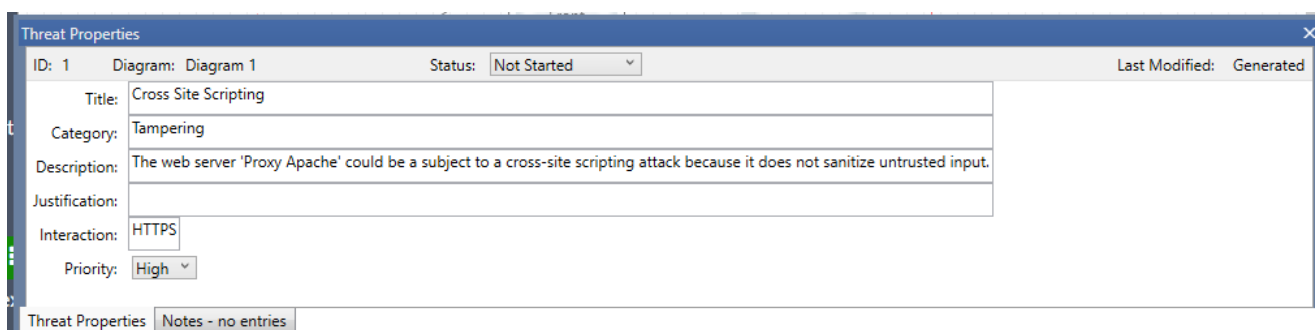


Figura 10 – Janela de propriedades da ameaça

A janela de propriedades da ameaça permite que o operador do TMT edite as informações dos campos, com exceção do campo interação. Deste modo, o analista pode



alterar as informações desejadas, inclusive o nível de prioridade da ameaça.

O quarto passo é aplicar o *framework* DREAD em cada ameaça identificada pelo TMT, para calcular o risco e assim classificá-la entre prioridade alta, média ou baixa. Para isso é necessário responder com a escala de 0-10 uma pergunta para cada letra do acrônimo. As perguntas e os marcos na escala em cada categoria utilizadas neste estudo foram as descritas em [Ransome et al. \(2014\)](#), e são as seguintes:

- **Dano potencial:** Se ocorrer a exploração da ameaça, quanto dano isso causará?
  - 0 = nenhum;
  - 5 = Apenas o usuário em questão será afetado;
  - 10 = Todo o sistema será destruído ou todos os dados serão comprometidos.
- **Reproduzibilidade:** Quão fácil é reproduzir a exploração desta ameaça?
  - 0 = muito difícil ou impossível de reproduzir, mesmo para administradores do aplicativo;
  - 5 = um ou dois passos são necessários, mas é preciso ser um usuário autorizado;
  - 10 = só um web *browser* e endereço é suficiente, sem autenticação.
- **Explorabilidade:** O que é necessário para explorar esta ameaça?
  - 0 = conhecimento avançado em programação e redes, com ferramentas de ataque personalizadas ou avançadas;
  - 5 = *malware* existe na internet ou uma exploração é facilmente realizada usando as ferramentas de ataque disponíveis;
  - 10 = só um navegador web.
- **Usuários afetados:** Quantos usuários serão afetados?
  - 0 = nenhum;
  - 5 = alguns, mas não todos;
  - 10 = todos.
- **Descoberta:** Quão fácil é descobrir esta ameaça?
  - 0 = muito difícil a impossível, requer código-fonte ou acesso administrativo;
  - 5 = pode descobrir adivinhando ou monitorando traços de rede;
  - 9 = detalhes de falhas como esta já estão no domínio público e podem ser facilmente descobertos usando um mecanismo de pesquisa;

- 10 = as informações estão visíveis na barra de endereços do navegador web ou em um formulário.

As repostas de 0 a 3 são consideradas de baixa prioridade, de 4 a 6 de média prioridade e de 7 a 10 de alta prioridade. Com estas informações preencheremos a matriz apresentada na Tabela 2, onde o baixo risco vale 1, o médio vale 2 e o alto risco vale 3 pontos. Os pontos de todas as categorias são somados para chegar a pontuação total da ameaça analisada, se a pontuação final estiver entre 12 e 15 é considerada de alto nível, entre 8 e 11 é média e entre 5 e 7 é de baixo risco. Desta forma, as ameaças são priorizadas e se finaliza a coleta de dados da modelagem de ameaças.

## 5.2 Dados coletados da modelagem de ameaças

A partir do diagrama apresentado na Figura 9, o software *Microsoft Thread Modeling Tool* devolveu 55 ameaças. Estas ameaças foram priorizadas seguindo o modelo de risco DREAD e organizados na tabela 7 do apêndice A.

### 5.2.1 Dados do DREAD

Por meio do DREAD as ameaças encontradas pelo TMT foram priorizadas e documentadas por ordem de prioridade na Tabela 3.

Tabela 3 – Priorização realizada com o DREAD

ID	Ameaças	D	R	E	A	D	Soma	Prioridade
34	Falsificando a entidade externa do usuário humano	3	1	3	3	2	12	Alta
44	Falha ou parada potencial do processo para API	3	2	3	3	1	12	Alta
45	HTTPS de fluxo de dados é potencialmente interrompido	3	1	3	3	3	13	Alta
54	Falha ou parada potencial do processo para API	3	2	3	3	1	12	Alta
64	Falha ou parada potencial do processo para API	3	2	3	3	1	12	Alta
65	HTTPS de fluxo de dados é potencialmente interrompido	3	1	3	3	3	13	Alta
14	Memória de processo <i>Proxy Apache</i> adulterada	3	1	3	2	2	11	Média

Continua na próxima página

ID	Ameaças	D	R	E	A	D	Soma	Prioridade
17	<i>Script</i> entre sites	2	2	1	1	3	9	Média
20	Memória de processo <i>Middleware</i> adulterada	3	1	3	2	2	11	Média
22	Elevação usando representação	3	1	3	3	1	11	Média
25	Potencial repúdio de dados no <i>Proxy Apache</i>	2	2	1	2	2	9	Média
26	Falha ou parada de processo potencial para <i>proxy Apache</i>	3	1	1	3	1	9	Média
27	Fluxo de dados HPPTS potencialmente interrompido	3	1	3	3	1	11	Média
29	Elevação alterando o fluxo de execução no <i>Proxy Apache</i>	3	1	3	3	1	11	Média
30	Falsificação de solicitação entre sites	2	2	1	1	3	9	Média
31	Falsificação da entidade de destino externo do usuário humano	3	2	2	2	1	10	Média
32	Usuário humano de entidade externa potencialmente nega o recebimento de dados	3	1	3	3	1	11	Média
33	Fluxo de dados HPPTS potencialmente interrompido	3	1	3	3	1	11	Média
36	Elevação usando representação	3	1	3	3	1	11	Média
37	Elevação usando representação	3	1	3	3	1	11	Média
39	Falsificação da entidade de destino externo do CFM da API	3	1	2	2	3	11	Média
40	API de entidade externa CFM potencialmente nega o recebimento de dados	2	2	1	2	1	8	Média
41	HTTPS de fluxo de dados é potencialmente interrompido	3	1	3	3	1	11	Média
42	Falsificando a entidade externa CFM da API	2	2	1	2	1	8	Média
43	Potencial repúdio de dados pela API	2	2	1	2	1	8	Média
46	Elevação usando representação	3	2	2	2	2	11	Média

Continua na próxima página

ID	Ameaças	D	R	E	A	D	Soma	Prioridade
47	API pode estar sujeita a elevação de privilégio usando execução remota de código	1	3	2	1	2	9	Média
48	Elevação alterando o fluxo de execução na API	3	1	3	1	3	11	Média
49	Falsificação da entidade de destino externo CRF da API	3	1	2	2	3	11	Média
50	CRF da API de entidade externa potencialmente nega o recebimento de dados	2	2	1	2	1	8	Média
51	HTTPS de fluxo de dados é potencialmente interrompido	3	1	3	3	1	11	Média
52	Falsificação da entidade externa CRF da API	2	2	1	2	1	8	Média
53	Potencial repúdio de dados pela API	2	2	1	2	1	8	Média
55	HTTPS de fluxo de dados é potencialmente interrompido	3	1	3	3	1	11	Média
56	Elevação usando representação	3	2	2	2	2	11	Média
57	API pode estar sujeita a elevação de privilégio usando execução remota de código	1	3	2	1	2	9	Média
58	Elevação alterando o fluxo de execução na API	3	1	3	1	3	11	Média
59	Falsificação da entidade de destino externo CRO da API	3	1	2	2	3	11	Média
60	CRO de API de entidade externa potencialmente nega o recebimento de dados	2	2	1	2	1	8	Média
61	HTTPS de fluxo de dados é potencialmente interrompido	3	1	3	3	1	11	Média
62	Falsificação da entidade externa CRO da API	2	2	1	2	1	8	Média
63	Potencial repúdio de dados pela API	2	2	1	2	1	8	Média
66	Elevação usando representação	3	2	2	2	2	11	Média

Continua na próxima página

ID	Ameaças	D	R	E	A	D	Soma	Prioridade
67	API pode estar sujeita a elevação de privilégio usando execução remota de código	1	3	2	1	2	9	Média
68	Elevação alterando o fluxo de execução na API	3	1	3	1	3	11	Média
15	<i>Script</i> entre sites	2	2	1	1	1	7	Baixa
28	<i>Proxy Apache</i> pode estar sujeito a elevação de privilégio usando a execução remota de código	1	3	1	1	1	7	Baixa
1	<i>Script</i> entre sites	1	2	1	1	1	6	Baixa
2	Elevação usando representação	1	1	1	1	1	5	Baixa
16	Elevação usando representação	1	1	1	1	1	5	Baixa
18	Elevação usando representação	1	1	1	1	1	5	Baixa
19	Elevação usando representação	1	1	1	1	1	5	Baixa
21	<i>Script</i> entre sites	1	2	1	1	1	6	Baixa
35	Elevação usando representação	1	1	1	1	1	5	Baixa
38	Elevação usando representação	1	1	1	1	1	5	Baixa

### 5.3 Análise dos dados obtidos

Para facilitar o processo de análise dos dados obtidos foram utilizados os seguintes gráficos:

- Gráfico de contagem de ameaças por categoria (Figura 11);
- Gráfico de contagem de interação por tipo (Figura 12);
- Gráfico de contagem de ameaças por categoria (Figura 13);
- Gráfico de contagem de ameaças por categoria e por severidade (Figura 14).

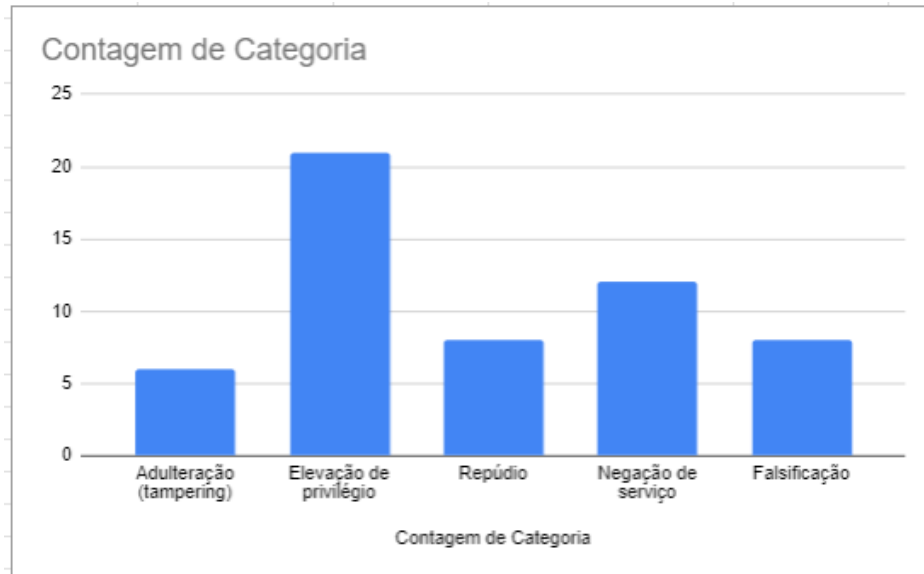


Figura 11 – Gráfico de contagem de ameaças por categoria

Como demonstrado pelo gráfico acima e seguindo o modelo STRIDE da *Microsoft* as 55 ameaças identificadas pelo TMT foram classificadas da seguinte maneira:

- 8 são referentes a Falsificação (*Spoofing*);
- 6 são relacionadas a Adulteração (*Tampering*);
- 8 são alusivas ao Repúdio (*Repudiation*);
- 0 referente ao Divulgação de informação (*Information Disclosure*);
- 12 ameaças concernente a Negação de serviço (*Denial Of Service*);
- 21 são relacionadas a Elevação de privilégio (*Elevation Of Privilege*).

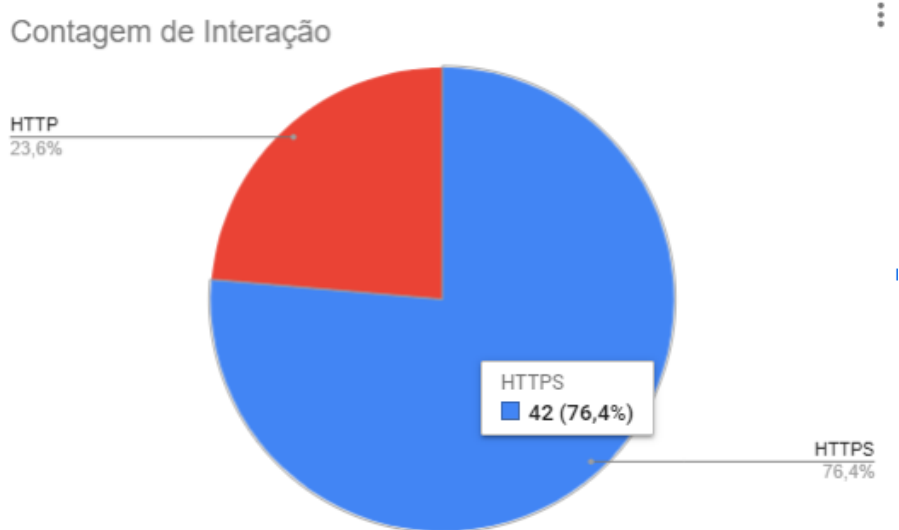


Figura 12 – Gráfico de contagem de interações por tipo

A Figura 12 retrata que das 55 ameaças indicadas pelo TMT, 42 são provenientes das interações *Hyper Text Transfer Protocol Secure* (HTTPS) e 13 são de interações *Hypertext Transfer Protocol* (HTTP).

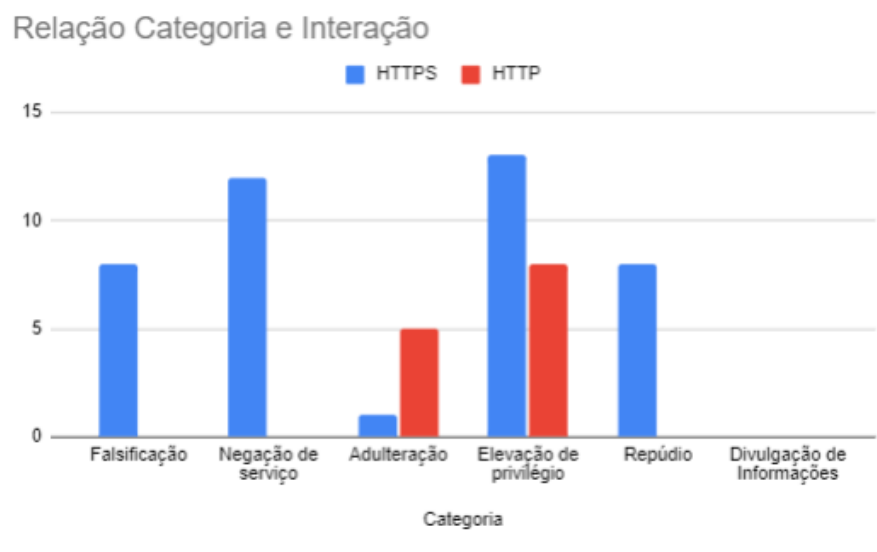


Figura 13 – Gráfico de contagem de ameaças por categoria

A Figura 13 ilustra a relação entre a categoria e interação das ameaças encontradas. Destas foram identificadas as informações apresentadas na Tabela 4.

Categoria	Porcentagem da interação HTTPS	Porcentagem da interação HTTP
Falsificação	100%	0%
Negação	100%	0%
Adulteração	16,67%	83,33%
Elevação de privilégio	61,90%	38,1%
Repúdio	100%	0%

Tabela 4 – Dados coletados na modelagem de ameaças

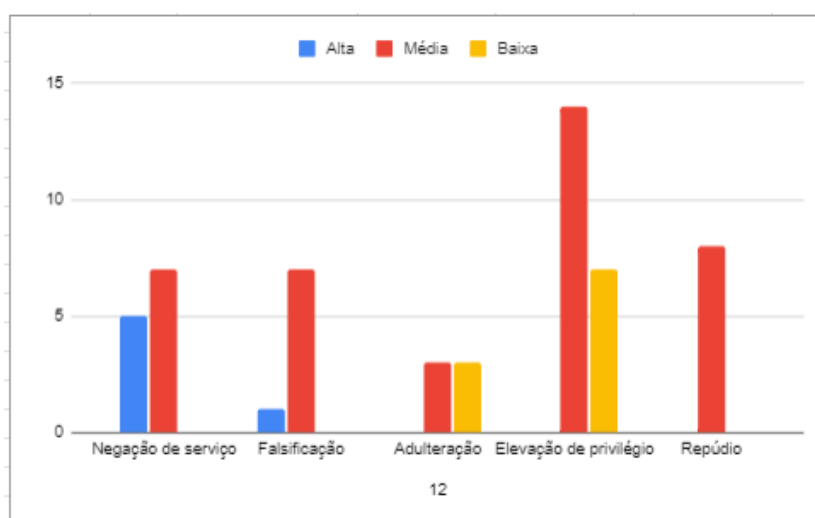


Figura 14 – Gráfico de contagem de ameaças por categoria e por severidade

A Figura 14 demonstra a relação das severidades encontradas por categoria. Os números estão descritos na Tabela 5.

Categoria	Severidade alta	Severidade média	Severidade baixa	Total
Falsificação	1	7	0	8
Negação de serviço	5	7	0	12
Adulteração	0	3	3	6
Elevação de privilégio	0	14	7	21
Repúdio	0	8	0	8
Total	6	38	11	55

Tabela 5 – Dados coletados na modelagem de ameaças



## 6 Resultados do Teste Estático de Segurança

O presente capítulo descreve o passo-a-passo executado neste estudo de caso para coletar os dados do teste estático de segurança (SAST), usando o *SonarQube*. Na sequência, os dados coletados são analisados e dissecados empregando tabelas e gráficos.

### 6.1 Passo-a-passo da coleta de dados

O teste estático de segurança foi efetuado por meio da versão gratuita do *SonarQube*, o *SonarQube Community Edition*. Para baixar esta ferramenta basta acessar o site<sup>1</sup> do mesmo, e selecionar a versão que deseja, a *Community Edition* que é gratuita ou a *Enterprise Edition* a paga, destinada a empresas.

O primeiro passo para utilizar o *SonarQube* é configurá-lo na máquina que o rodará. Para isso, é necessário ter o Java 17 ou superior instalado. Descompactar o arquivo baixado do software e colocá-lo no diretório raiz. Dentro da pasta da ferramenta, o usuário deverá entrar na pasta *conf*, abrir o arquivo *wrapper.conf* no editor de texto da sua preferência, atualizar o valor da variável *Wrapper.java.command* como o caminho para o Java na máquina em questão, exemplo: “C:\ProgramFiles\Java\jdk – 11.0.8\bin\java”.

No terminal de comando, deve ser aberta a pasta da ferramenta, entrar no diretório *bin*, acessar a pasta do sistema operacional que roda no computador em questão e digitar “*StartSonar.bat*”. Com isso, o Sonar será ativado e rodará na porta 9000. Para usá-lo basta digitar na barra de navegação do *brower* “*localhost:9000*”.

O acesso inicial requer o uso do nome de usuário e senha ‘*admin*’. Após iniciar a sessão, é possível modificar a senha. Para começar a utilizar o SAST, o primeiro passo consiste em criar um novo projeto. Para isso, é necessário selecionar o método de análise e fazer as configurações da análise. As opções de método de análise são: com o *Jenkins*, o *GitHub Actions*, *Bitbucket Pipelines*, *GitLab CI*, *Azure Pipelines*, outro CI ou local. O presente estudo foi realizado pelo método Local e com as configurações padrão de análise oferecidas pela ferramenta. Após isto, deve-se criar uma chave única e um nome descritivo para o projeto.

A análise de projeto é iniciada com a geração de um *token*, para gerar o *token* deve ser inserido um nome para o mesmo. A própria ferramenta informa que este *token* é usado para te identificar quando a análise é realizada e se tiver sido comprometido você pode revogá-lo a qualquer momento da sua conta de usuário. O segundo passo antes de iniciar a análise é escolher qual a linguagem do projeto que será analisado. O *SonarQube*

<sup>1</sup> <https://www.sonarsource.com/products/sonarqube/>

apresenta quatro opções, *Maven*, *Gradle*, *.NET* e *Outros (JS, TS, GO, Python, PHP, ...)*. Após selecionada a linguagem serão devolvidos os comandos necessários para rodar a ferramenta. Se a opção escolhida for ‘Outros’ também será necessário baixar a ferramenta *Sonar Scanner* e adicioná-la ao *path* das variáveis de ambiente.

O passo seguinte é abrir o terminal de comando na pasta do projeto e colar os comandos passados pelo *SonarQube*. Desta maneira a análise será realizada e os resultados serão mostrados no “*localhost:9000*”.

É importante ressaltar que, devido a um conflito de versões do Java entre o *SonarQube* e a API, foi necessário utilizar a versão 8.9 da ferramenta SAST na análise da API, enquanto nas demais análises foi utilizada a versão 10.6.

## 6.2 Dados coletados do teste de segurança estático

No presente estudo, o SAST foi realizado em três partes, uma com a API do VALIDAR, outra com o *middleware* e outra com o *front-end*. Para a API, a opção escolhida foi o Java com o *Maven*, para o *middleware* e o *front-end* a opção escolhida foi Outros.

Para um melhor entendimento dos resultados apresentados, é importante conhecer alguns conceitos fundamentais. Entre esses conceitos estão os *code smells*, também conhecidos como maus cheiros de código, e os pontos de acesso de segurança.

Os *code smells* são, como definido pelo próprio *Sonar Source*, categorias de problemas no código, como código duplicado, métodos muito longos, uso excessivo de comentários para explicar o código. Estes podem sinalizar problemas profundos a longo prazo, ou seja, funcionam como indicadores de alerta. Os maus cheiros de código surgem devido a práticas de codificação inadequadas.

Os “pontos críticos de segurança”, ou *hotspots*, mencionados pelo *SonarQube* referem-se a códigos sensíveis à segurança. Nessas situações, é crucial realizar uma revisão, pois podem representar vulnerabilidades. Assim, é necessário corrigir esses pontos para proteger o código.

É crucial compreender a diferença entre um ponto crítico de segurança e uma vulnerabilidade. A principal distinção reside na necessidade de revisão antes de decidir pela aplicação de uma correção:

- Um ponto crítico de segurança não compromete a segurança geral da aplicação por si só. Cabe ao desenvolvedor revisar o código para determinar se uma correção é necessária para garantir a proteção do código.
- Uma vulnerabilidade, por outro lado, é um problema que afeta diretamente a segurança do software e deve ser corrigido imediatamente ao ser identificado.

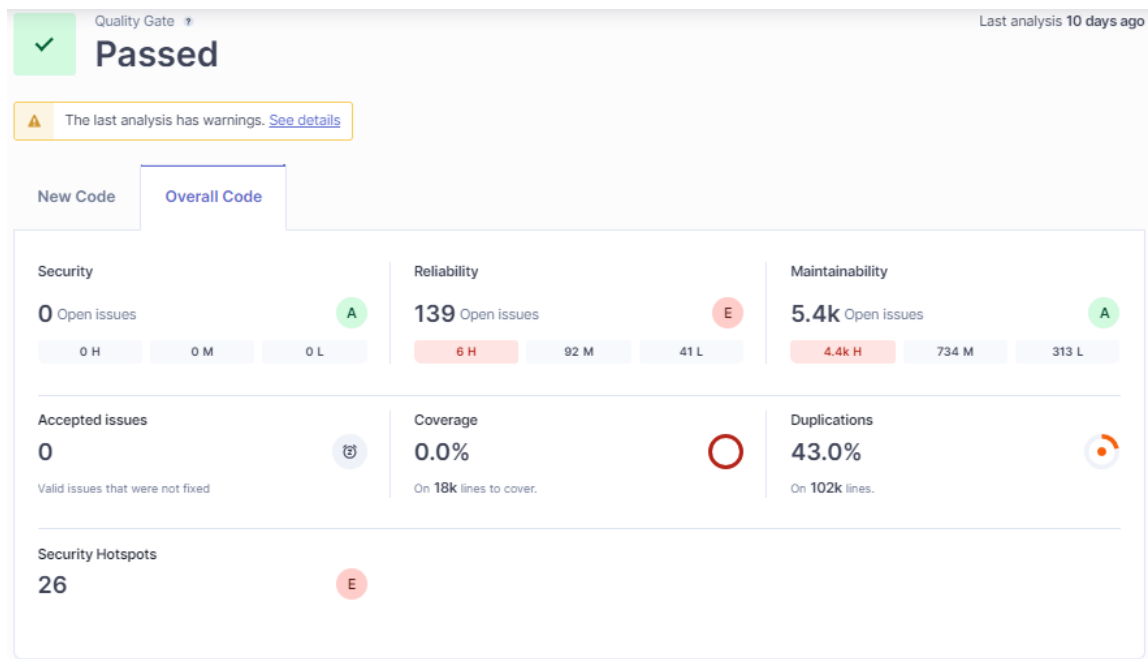


Figura 15 – Resultado do SAST para o *front-end* do Validar.

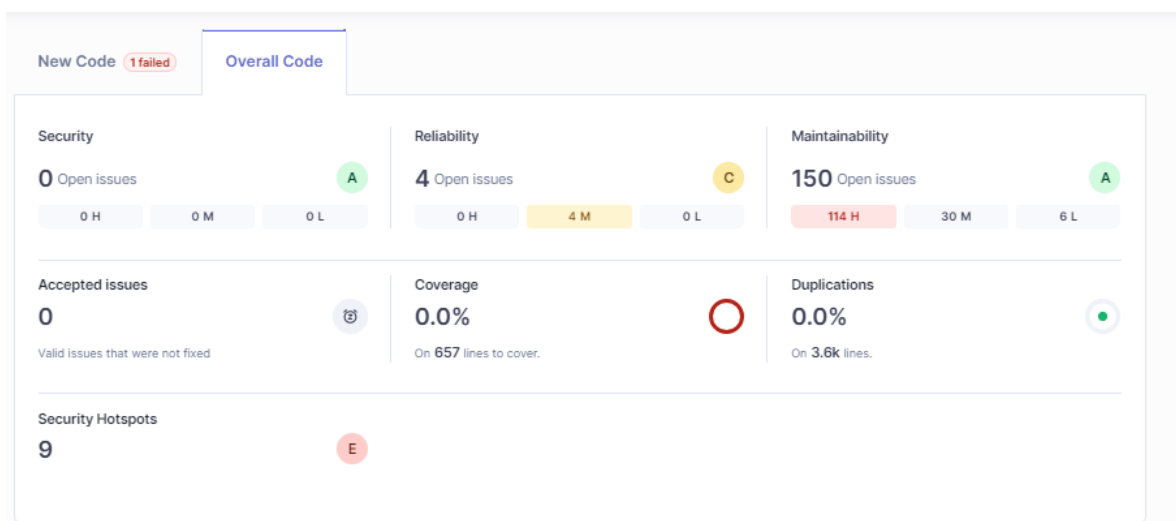


Figura 16 – Resultado do SAST para o *middleware* do Validar.

### 6.2.1 Dados do *front-end* do Validar

A Figura 15 apresenta o resultado da análise do Sonar para o *front-end* do Validar. Foram identificados 139 problemas de confiabilidade, 5.400 problemas de manutenibilidade e 26 pontos críticos de segurança. Não há problemas de segurança ou vulnerabilidades abertas. A cobertura de testes é de 0.0%, e o código possui 43.0% de duplicação em 102.000 linhas.

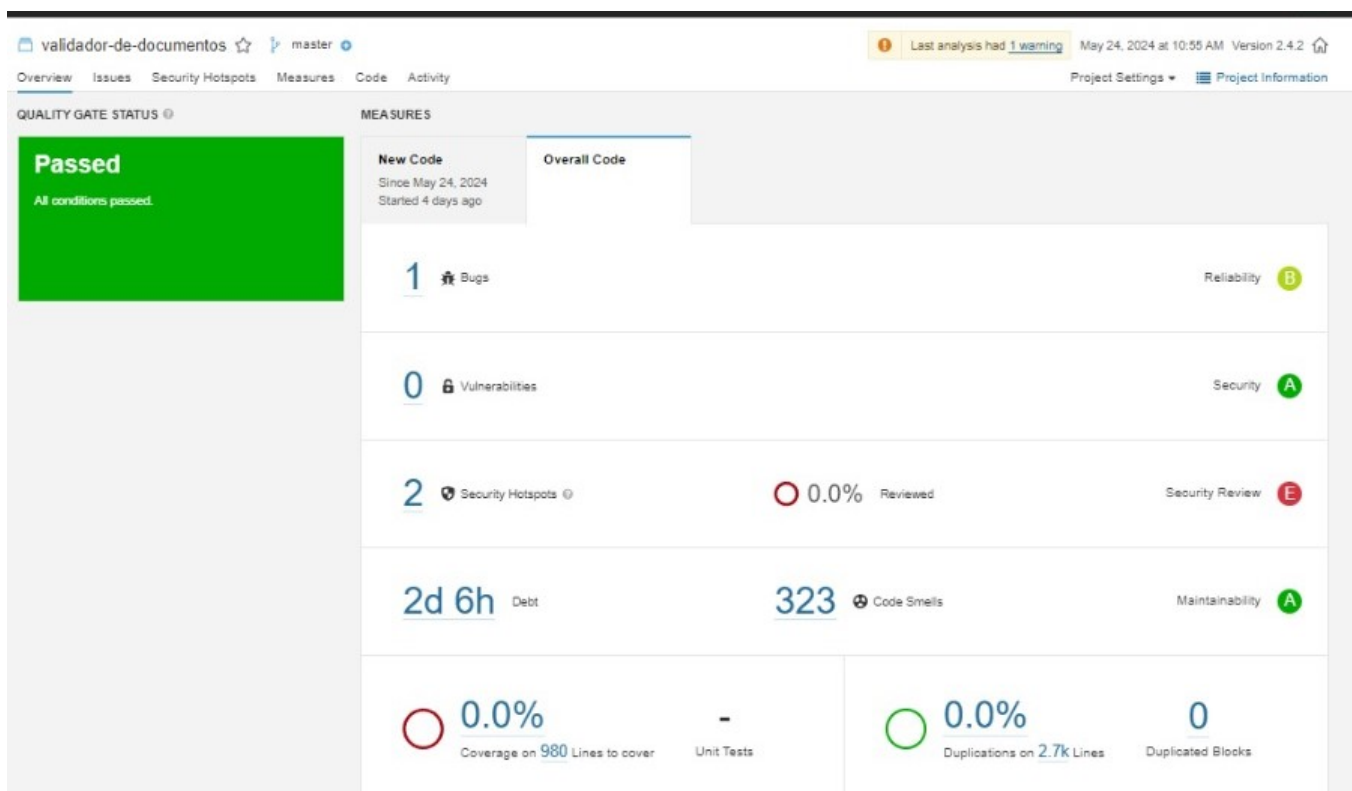


Figura 17 – Resultado do SAST para a API do Validar.

## 6.2.2 Dados do *middleware* do Validar

A Figura 16 apresenta o resultado da análise do Sonar para o *middleware* do Validar. O projeto possui 4 problemas de confiabilidade e 150 problemas de manutenibilidade, além de 9 pontos críticos de segurança. Não há problemas de segurança ou vulnerabilidades abertas. A cobertura de testes é de 0.0%, e o código não possui duplicação em 3.600 linhas.

## 6.2.3 Dados da API do Validar

A Figura 17 apresenta o resultado da análise do Sonar para a API do Validar. Nas 2.700 linhas de código da API do validador, onde o núcleo de verificação é uma biblioteca externa, foram encontrados 323 maus cheiros de código, 2 pontos críticos de segurança e apenas 1 bugs. Não foram encontradas vulnerabilidades, ou código duplicado.

## 6.3 Análise dos dados obtidos

Ao realizar o teste de segurança estático, não foram identificadas vulnerabilidades propriamente ditas. Portanto, a análise será iniciada pelos pontos críticos de segurança,

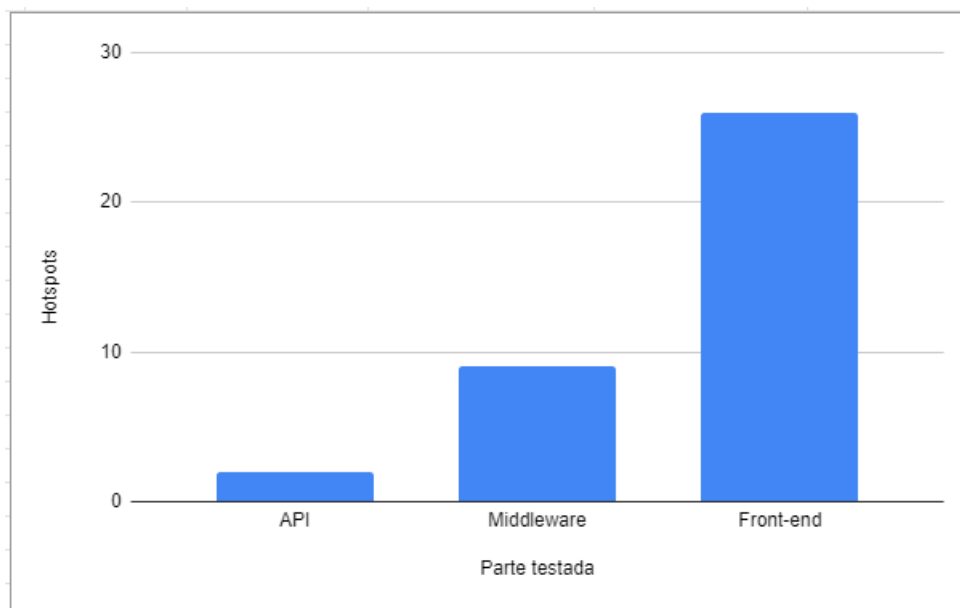


Figura 18 – Quantidade de pontos de acesso de segurança por parte testada.

pois estes podem eventualmente se tornar vulnerabilidades no futuro. A Figura 18 ilustra graficamente a quantidade de pontos críticos de segurança encontrados por parte testada.

### 6.3.1 Análise dos dados da API do Validar

Os dois *hotspots* encontrados na API foram:

1. Certifique-se de que o uso deste endereço IP codificado seja seguro aqui, representado no Figura 19.
2. Certifique-se de que esta não seja uma credencial codificada, ilustrado na Figura 20.

O *hotspot* “Certifique-se de que o uso deste endereço IP codificado seja seguro aqui” na realidade não faz referência a um endereço de IP, mas ao OID 2.16.76.1 definido na documento "ATRIBUIÇÃO DE OID NA ICP-BRASIL (DOC-ICP-04.01) Versão 3.3"<sup>2</sup>. Este documento traz as seguinte informações, O OID (*Object Identifier*) é um número único que identifica classes de objetos ou atributos em diretórios, garantindo unicidade global. O Brasil recebeu o OID-raiz 2.16.76.1 da ISO, e a partir desse OID, o ITI criou identificadores para cada Autoridade Certificadora (AC), Políticas de Certificados e outros elementos da ICP-Brasil, assegurando a identificação única de certificados, titulares e ACs emittentes.

O OID é uma informação pública e divulgada pelo próprio ITI, não representando uma vulnerabilidade e que faz desse ponto crítico de segurança um falso positivo.

<sup>2</sup> [https://mailman.iti.gov.br/images/repositorio/legislacao/documentos-principais/04.1/DOC-ICP-04.01\\_v.4.0\\_ATRIBUICAO\\_DE\\_OID\\_NA\\_ICP\\_-\\_BRASIL.pdf](https://mailman.iti.gov.br/images/repositorio/legislacao/documentos-principais/04.1/DOC-ICP-04.01_v.4.0_ATRIBUICAO_DE_OID_NA_ICP_-_BRASIL.pdf)

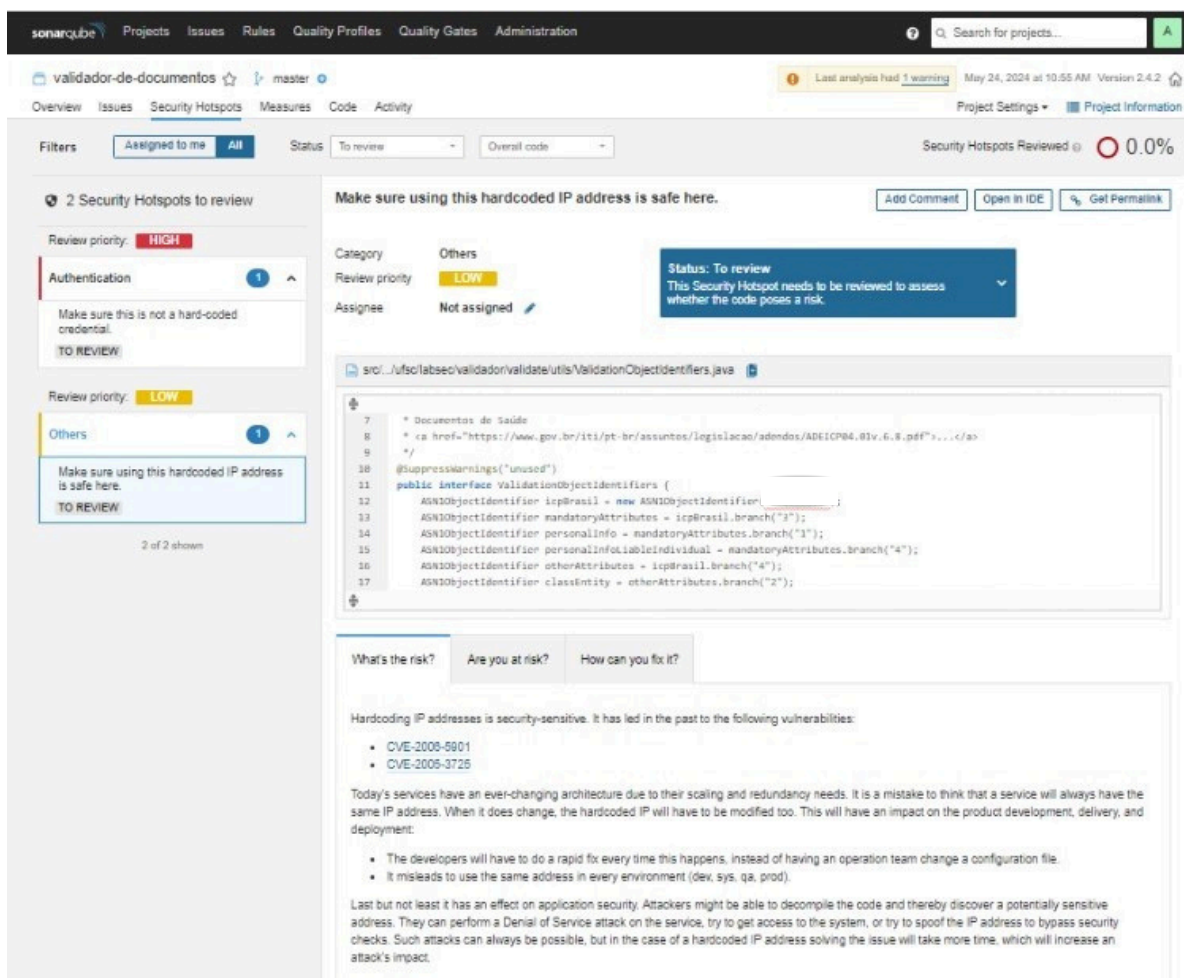


Figura 19 – Primeiro ponto crítico de segurança encontrado da API.

The screenshot displays the SonarQube interface for a project named 'validador-de-documentos'. It shows two security hotspots to be reviewed. The first hotspot, with a 'HIGH' review priority, is titled 'Make sure this is not a hard-coded credential' and is categorized as 'Authentication'. The code snippet shows a Maven configuration file (pom.xml) with a hard-coded Sonar login and password. The status is 'To review' with a note: 'This Security Hotspot needs to be reviewed to assess whether the code poses a risk.' The interface also includes a 'What's the risk?' section explaining the danger of hard-coded credentials and a 'How can you fix it?' section recommending the use of external services for secrets.

Figura 20 – Segundo crítico de segurança encontrado da API.

O *hotspot*, Certifique-se de que esta não seja uma credencial codificada, apresentado na Figura 20 foi causado pela configuração do sonar e portanto não representa uma possível vulnerabilidade, uma vez que não faz parte do código oficial.

### 6.3.2 Análise dos dados do *middleware* do Validar

O *SonarQube* identificou 9 *hotspots* no *middleware* do objeto de análise deste estudo. Foram encontrados 2 pontos críticos com uma necessidade média de revisão, 7 com necessidade baixa de revisão, como demonstrado na Figura 21.

Os pontos críticos de segurança com média necessidade de revisão são:

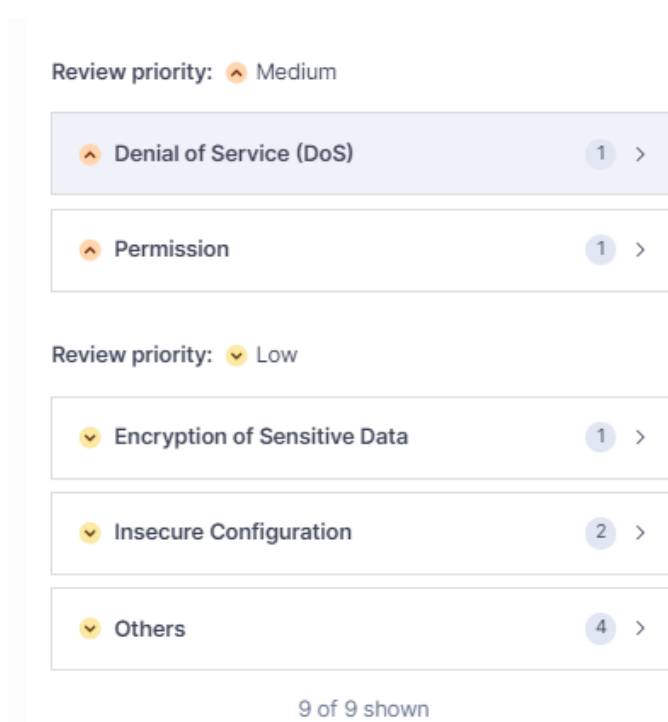


Figura 21 – Classificação de severidade dos pontos de acesso de segurança encontrados no **middleware**

- **Negação de serviço (DoS)**

1. Certifique-se de que o limite de comprimento do conteúdo seja seguro aqui (Figura 22).

- **Permissão**

1. A imagem do nó é executada com *root* como usuário padrão (Figura 23).

Os pontos críticos de segurança com baixa necessidade de revisão são:

- **Criptografia de dados confidenciais**

1. Usar o protocolo *http* é inseguro. Use *https* em vez disso (Figura 24).

- **Configuração insegura**

1. Certifique-se de que a ativação do CORS seja segura aqui (Figura 25).
2. Certifique-se de que a ativação do CORS seja segura aqui (Figura 26).

- **Outros**

1. A omissão de *-ignore-scripts* pode levar à execução de *scripts shell* (Figura 27).



Make sure the content length limit is safe here. [🔗](#)

Allowing requests with excessive content length is security-sensitive [javascript:S5693](#)

Status: To review

This security hotspot needs to be reviewed to assess whether the code poses a risk. [Review](#)

Where is the risk? | What's the risk? | Assess the risk | How can I fix it? | Activity

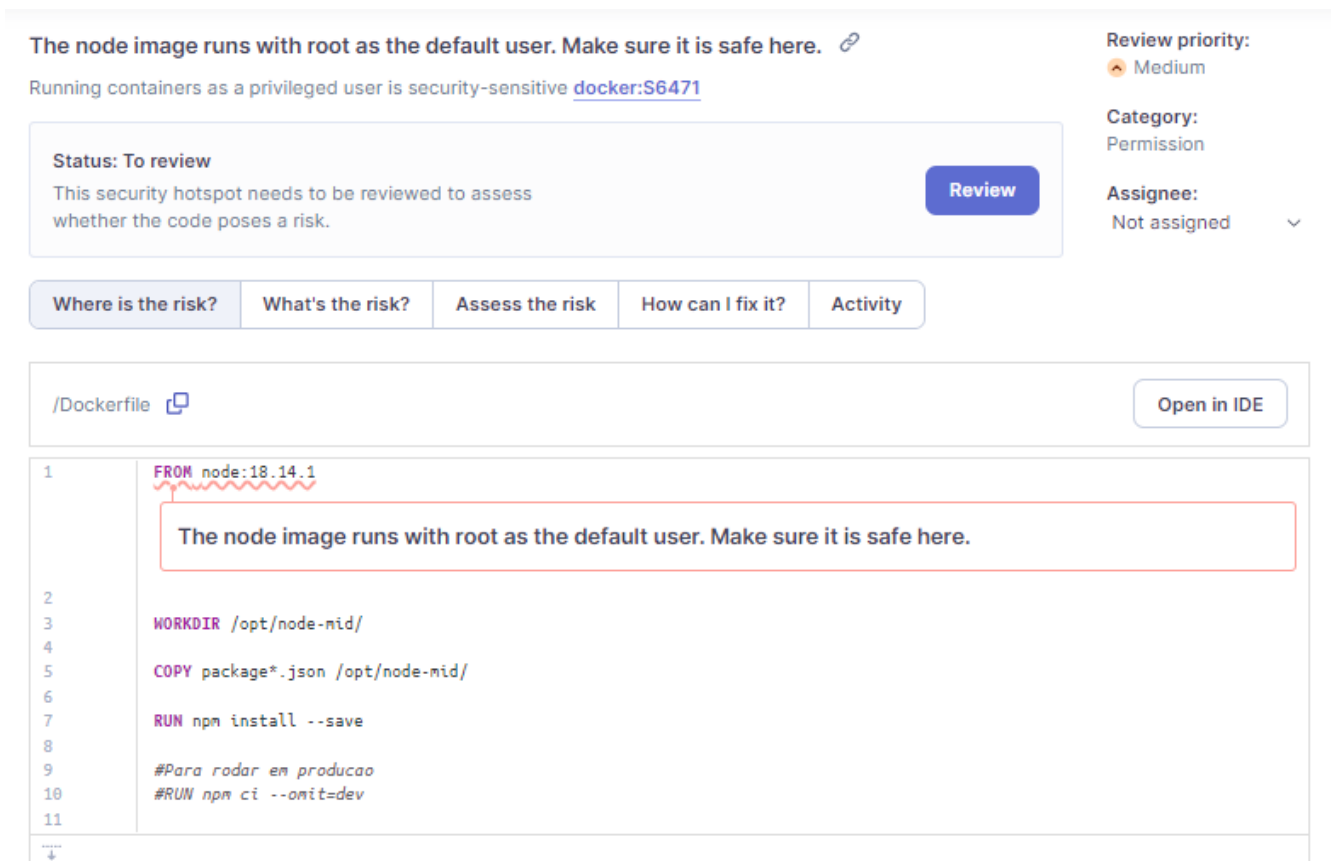
src/routes/routes.js [📄](#)

```
3   const multer = require('multer');
4   const axios = require('axios');
5
6   const simples = require('../tratamentos/simples');
7   const conformidade = require('../tratamentos/conformidade/conformidade');
8   const pdf = require('../tratamentos/conformidade/criarPdf');
9   const geral = require('../tratamentos/geral');
10  const config = require('./config')
11
12  // Configuração para limitar o tamanho do payload de dados binários para 50MB
13  const storage = multer.memoryStorage();
```

Make sure the content length limit is safe here.

Figura 22 – Primeiro ponto de acesso de segurança encontrados no **middleware**

2. Este *framework* implicitamente divulga informações de versão por padrão (Figura 28).
3. Certifique-se de não usar `rel="noopener"` é seguro aqui (Figura 29).
4. Certifique-se de não usar `rel="noopener"` é seguro aqui (Figura 30).



The screenshot displays a security tool interface. At the top, a message reads: "The node image runs with root as the default user. Make sure it is safe here." with a link icon. Below this, it says "Running containers as a privileged user is security-sensitive [docker:S6471](#)".

On the right side, there are fields for "Review priority: Medium", "Category: Permission", and "Assignee: Not assigned".

The main area shows a code editor for a Dockerfile. The code is as follows:

```
1 FROM node:18.14.1
2
3 WORKDIR /opt/node-mid/
4
5 COPY package*.json /opt/node-mid/
6
7 RUN npm install --save
8
9 #Para rodar em producao
10 #RUN npm ci --omit=dev
11
```

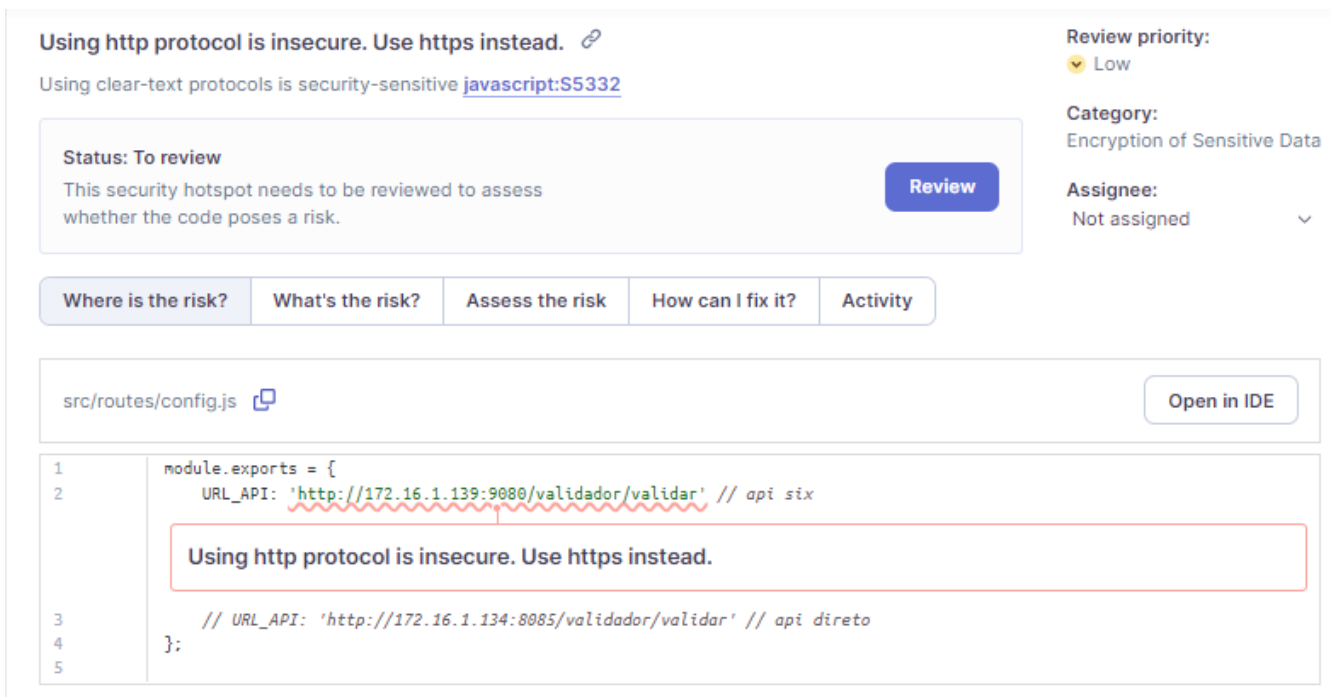
A red box highlights the first line, "FROM node:18.14.1", with the same security message: "The node image runs with root as the default user. Make sure it is safe here." Below the code editor, there are tabs for "Where is the risk?", "What's the risk?", "Assess the risk?", "How can I fix it?", and "Activity".

Figura 23 – Segundo ponto de acesso de segurança encontrados no **middleware**

### 6.3.2.1 Ponto crítico de segurança de negação de serviço (DoS) encontrados no *middleware*

O ponto crítico de segurança “Certifique-se de que o limite de comprimento do conteúdo seja seguro aqui”, ilustrado na Figura 22, possui um nível de severidade de baixa a moderada e pertence à categoria Negação de serviço (DoS). Isso ocorre porque os arquivos carregados são mantidos na memória. Embora exista uma delimitação de tamanho configurada para os arquivos, ainda é importante estar ciente das implicações de armazenar arquivos na memória, especialmente em um ambiente de produção onde múltiplos *uploads* simultâneos podem ocorrer.

As implicações de armazenar arquivos na memória incluem um possível esgotamento da RAM, especialmente com múltiplos *uploads* simultâneos, resultando em instabilidade ou travamentos do sistema devido aos limites físicos da memória (SOUZA, 2022). Isso degrada o desempenho, pois o consumo excessivo de memória diminui a velocidade de resposta e força o uso de memória virtual (*swap*), que é significativamente mais lenta (MARIJAN, 2023). A escalabilidade é comprometida, pois adicionar mais memória não é uma solução sustentável e é limitada pelo hardware disponível.



The screenshot displays a security tool interface. At the top, a header reads "Using http protocol is insecure. Use https instead." with a link icon. Below this, a sub-header states "Using clear-text protocols is security-sensitive" followed by a link "javascript:S5332".

The main content area is divided into two sections. On the left, a box contains the status "Status: To review" and a description: "This security hotspot needs to be reviewed to assess whether the code poses a risk." A blue "Review" button is positioned to the right of this text. On the right side, there are three fields: "Review priority:" set to "Low" (indicated by a yellow heart icon), "Category:" set to "Encryption of Sensitive Data", and "Assignee:" set to "Not assigned" with a dropdown arrow.

Below these sections is a navigation bar with five tabs: "Where is the risk?", "What's the risk?", "Assess the risk", "How can I fix it?", and "Activity".

The bottom section shows a code editor for the file "src/routes/config.js" with a copy icon and an "Open in IDE" button. The code is as follows:

```
1 module.exports = {
2   URL_API: 'http://172.16.1.139:9080/validador/validar' // api six
3   // URL_API: 'http://172.16.1.134:8085/validador/validar' // api direto
4 };
5
```

A red box highlights the first URL, with a tooltip that reads "Using http protocol is insecure. Use https instead."

Figura 24 – Terceiro ponto de acesso de segurança encontrados no **middleware**

### 6.3.2.2 Ponto crítico de segurança sobre permissão encontrados no *middleware*

O *hotspot* “A imagem do nó é executada com *root* como usuário padrão” (Figura 23), da categoria de Permissão, possui um nível de severidade médio. Executar contêineres como *root* aumenta significativamente o risco de ataques, pois um comprometimento do contêiner pode levar a um comprometimento do *host* ou de outros contêineres (ARTEM et al., ). Além disso, contêineres executados como *root* são mais atrativos para ataques maliciosos, pois podem executar comandos administrativos. Isso não só eleva o risco de segurança, mas também pode causar problemas de conformidade e expor a vulnerabilidades desnecessárias (ARTEM et al., ).

### 6.3.2.3 Ponto crítico de segurança sobre criptografia de dados confidenciais encontrados no *middleware*

O ponto crítico “Usar o protocolo HTTP é inseguro. Use HTTPS em vez disso” (Figura 24), relacionado à criptografia de dados confidenciais, possui um baixo nível de risco e não representa uma vulnerabilidade. Isto ocorre devido à presença de um *firewall* que protege o ambiente e restringe o acesso a usuários autorizados, além de os dados não serem sensíveis. No entanto, mesmo em ambientes protegidos por *firewall* e com acesso restrito, o uso de HTTPS é crucial para garantir a confidencialidade e integridade dos dados. HTTPS criptografa a transmissão, protegendo contra interceptações e modificações indesejadas, e verifica a identidade do servidor, garantindo que os dados sejam enviados

Make sure that enabling CORS is safe here. [🔗](#)

Having a permissive Cross-Origin Resource Sharing policy is security-sensitive [javascript:S5122](#)

**Status: To review**

This security hotspot needs to be reviewed to assess whether the code poses a risk. [Review](#)

**Review priority:** Low

**Category:** Insecure Configuration

**Assignee:** Not assigned

Where is the risk? | What's the risk? | Assess the risk | How can I fix it? | Activity

src/index.js [🔗](#) [Open in IDE](#)

```
3  const cors = require('cors');
4  const routes = require('./routes/routes');
5
6  const app = express();
7
8  // Configuração para limitar o tamanho do payload JSON, formulários HTML para 50MB
9  app.use(bodyParser.json({ limit: '50mb' }));
10 app.use(bodyParser.urlencoded({ limit: '50mb', extended: true }));
11
12 app.use(express.json());
13 app.use(cors());
```

Make sure that enabling CORS is safe here.

Figura 25 – Quarto ponto crítico de segurança encontrados no **middleware**

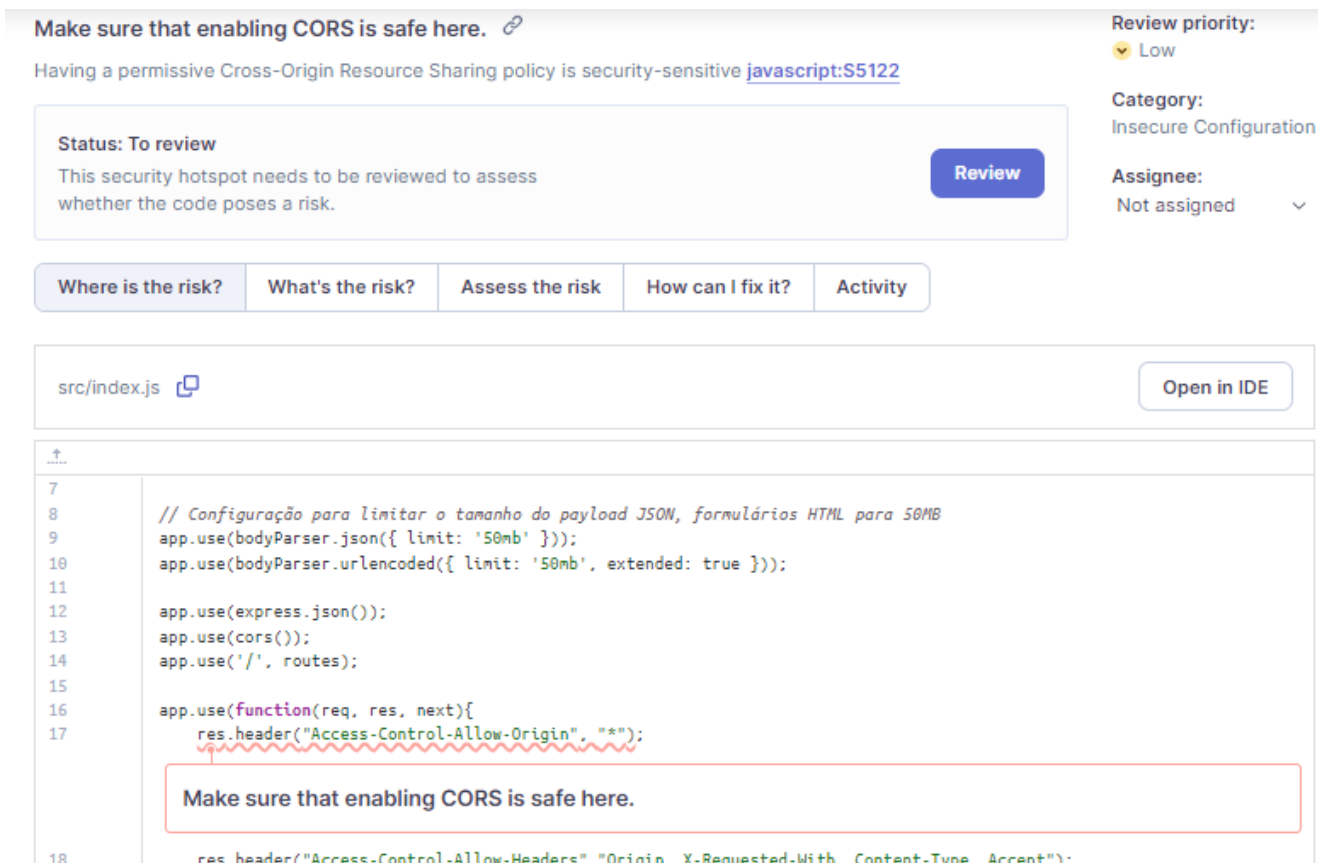
para o destinatário correto (ALVES, 2022).

#### 6.3.2.4 Pontos críticos de segurança de configuração insegura encontrados no *middleware*

Os pontos críticos de segurança quatro (Figura 25) e cinco (Figura 26), nomeados “Certifique-se de que a ativação do CORS seja segura aqui”, podem representar uma configuração insegura, mas possuem um baixo nível de severidade. Isso se deve ao fato de que, embora existam riscos associados a uma configuração permissiva de CORS, como a exposição de recursos, onde o acesso irrestrito permite que qualquer domínio acesse os recursos da API, potencialmente expondo dados sensíveis, esses riscos podem ser mitigados com configurações adequadas.

Riscos específicos associados ao CORS permissivo incluem facilitar ataques *cross-site*, como *Cross-Site Scripting (XSS)*, onde *scripts* maliciosos de outros domínios podem interagir com a API, e *Cross-Site Request Forgery (CSRF)*, permitindo que sites maliciosos façam requisições em nome do usuário sem consentimento, como demonstrado por Exploiting... (2023). Sem restrições adequadas, a segurança da API fica comprometida, permitindo uso indevido por terceiros, abuso de recursos e ataques de força bruta.

A severidade é considerada baixa destes pontos de acesso porque, embora os riscos



The screenshot displays a security tool interface. At the top, a header reads "Make sure that enabling CORS is safe here." with a link icon. Below this, a subtitle states: "Having a permissive Cross-Origin Resource Sharing policy is security-sensitive [javascript:S5122](#)".

The main content area shows a "Status: To review" box with the text: "This security hotspot needs to be reviewed to assess whether the code poses a risk." and a "Review" button.

On the right side, there are three fields: "Review priority:" set to "Low", "Category:" set to "Insecure Configuration", and "Assignee:" set to "Not assigned".

Below the status box is a navigation bar with five tabs: "Where is the risk?", "What's the risk?", "Assess the risk", "How can I fix it?", and "Activity".

The code editor shows the file "src/index.js" with a "Open in IDE" button. The code includes:

```
7
8 // Configuração para limitar o tamanho do payload JSON, formulários HTML para 50MB
9 app.use(bodyParser.json({ limit: '50mb' }));
10 app.use(bodyParser.urlencoded({ limit: '50mb', extended: true }));
11
12 app.use(express.json());
13 app.use(cors());
14 app.use('/', routes);
15
16 app.use(function(req, res, next){
17   res.header("Access-Control-Allow-Origin", "*");
18   res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
```

A red box highlights the line `res.header("Access-Control-Allow-Origin", "*");` with the text "Make sure that enabling CORS is safe here." overlaid on it.

Figura 26 – Quinto ponto crítico de segurança encontrados no **middleware**

sejam significativos, a configuração de CORS pode ser ajustada facilmente para mitigar esses riscos.

### 6.3.2.5 Outros pontos críticos de segurança de encontrados no *middleware*

O *hotspot* “A omissão de `-ignore-scripts` pode levar à execução de *scripts shell*” (Figura 27) possui um baixo nível de severidade. A execução de *scripts* não confiáveis com o comando `npm install` sem a flag `-ignore-scripts` permite a execução automática de *scripts* definidos no `package.json` de qualquer pacote, incluindo *scripts* maliciosos de pacotes comprometidos (LADISA et al., 2023). Isso pode resultar em vulnerabilidades de segurança, como a execução de código arbitrário e acesso não autorizado a recursos críticos do sistema. Embora a execução de *scripts* possa representar um risco significativo, é possível mitigar esse risco com uma configuração adequada e boas práticas de segurança. Por esse motivo, este *hotspot* é considerado de baixa severidade.

O *hotspot* “Este *framework* implicitamente divulga informações de versão por padrão” (Figura 28) indica que o uso do `Express.js` pode expor a versão do *framework*, tornando a aplicação suscetível a ataques que exploram vulnerabilidades específicas dessa versão. Ao permitir a identificação do *framework* e sua versão, atacantes podem direci-

**Omitting --ignore-scripts can lead to the execution of shell scripts. Make sure it is safe here.** [🔗](#)

Allowing shell scripts execution during package installation is security-sensitive [docker:S6505](#)

**Status: To review**

This security hotspot needs to be reviewed to assess whether the code poses a risk.

[Review](#)

**Review priority:** ▼ Low

**Category:** Others

**Assignee:** Not assigned ▼

Where is the risk?
What's the risk?
Assess the risk
How can I fix it?
Activity

/Dockerfile [🔗](#) [Open in IDE](#)

```

1 FROM node:18.14.1
2
3 WORKDIR /opt/node-mid/
4
5 COPY package*.json /opt/node-mid/
6
7 RUN npm install --save
    
```

Omitting --ignore-scripts can lead to the execution of shell scripts. Make sure it is safe here.

Figura 27 – Sexto ponto crítico de segurança encontrados no **middleware**

**This framework implicitly discloses version information by default. Make sure it is safe here.** [🔗](#)

Disclosing fingerprints from web application technologies is security-sensitive [javascript:S5689](#)

**Status: To review**

This security hotspot needs to be reviewed to assess whether the code poses a risk.

[Review](#)

**Review priority:** ▼ Low

**Category:** Others

**Assignee:** Not assigned ▼

Where is the risk?
What's the risk?
Assess the risk
How can I fix it?
Activity

src/index.js [🔗](#) [Open in IDE](#)

```

1 const express = require('express');
2 const bodyParser = require('body-parser');
3 const cors = require('cors');
4 const routes = require('./routes/routes');
5
6 const app = express();
    
```

This framework implicitly discloses version information by default. Make sure it is safe here.

Figura 28 – Sétimo ponto crítico de segurança encontrados no **middleware**

**Make sure not using rel="noopener" is safe here.** [Web:S5148](#)

Authorizing an opened window to access back to the originating window is security-sensitive [Web:S5148](#)

Status: To review  
This security hotspot needs to be reviewed to assess whether the code poses a risk. [Review](#)

Review priority: Low  
Category: Others  
Assignee: Not assigned

Where is the risk? What's the risk? Assess the risk? How can I fix it? Activity

src/internacional/trust-list-BR.html [Open in IDE](#)

```
64 <p class="tp1 nt-2">
65 O serviço de Validação de Assinaturas Eletrônicas requer, para a adequada operação, o registro de
66 âncoras de confiança. Essas âncoras são representadas por meio de Listas de Confiança cujo
67 conteúdo são informações sobre as Autoridades Certificadoras credenciadas e reconhecidas pelo
68 serviço, para a validação de assinaturas eletrônicas.
69
70 </p>
71 <p class="tp2">
72 Atualmente, o serviço de validação, reconhece as seguintes de Listas de Confiança, em formato
73 XML:
74 <p>
75 <a id="link"
76 href="https://honor-validar.iti.gov.br/internacional/trust-list-BR.xml"
77 target="_blank" download>
78 </a>
79 </p>
80 <p class="tp3">
```

Figura 29 – Oitavo ponto crítico de segurança encontrados no **middleware**

onar ataques conhecidos para essa versão específica, aumentando o risco de comprometimento da segurança. A documentação oficial<sup>3</sup> do *Express* sugere desativar o cabeçalho *X-Powered-By* para evitar a exposição da versão do *framework* e, assim, proteger a aplicação de ataques direcionados a vulnerabilidades específicas dessa versão. O nível de severidade dessa vulnerabilidade é baixo, pois embora exponha a aplicação a possíveis ataques, a mitigação é simples e eficaz.

Os pontos críticos de segurança oito (Figura 29) e nove (Figura 30), nomeados “Certifique-se de não usar *rel="noopener"* é seguro aqui” indica um risco de segurança relacionado à abertura de links em novas abas sem o uso de *rel="noopener"*, o que pode levar a ataques de *tabnabbing* e permitir manipulação da página de origem (OWASP, b). Embora a severidade desse risco seja baixa, a mitigação é simples e eficaz, consistindo na adição do atributo *rel="noopener"* aos links, garantindo a segurança contra possíveis redirecionamentos maliciosos e manipulações. Além disso, OWASP (b) afirma que os links que usam *target=\_\_blank* agora têm implícito *rel = "noopener"* em navegadores modernos, então essa vulnerabilidade não é tão disseminada e crítica quanto antes.

<sup>3</sup> <https://expressjs.com/en/advanced/best-practice-security.html>

**Make sure not using rel="noopener" is safe here.** [Web:S5148](#)

Authorizing an opened window to access back to the originating window is security-sensitive

**Status: To review**  
This security hotspot needs to be reviewed to assess whether the code poses a risk. [Review](#)

**Review priority:** Low

**Category:** Others

**Assignee:** Not assigned

Where is the risk? | What's the risk? | Assess the risk | How can I fix it? | Activity

src/internacional/trust-list-BR.html [Open in IDE](#)

```
72      Atualmente, o serviço de validação, reconhece as seguintes de Listas de Confiança, em formato
73      XML:
74      </p>
75      <a id="link"
76      href="https://homol-validar.iti.gov.br/internacional/trust-list-BR.xml"
77      target="_blank" download>
78      <p id="listaConfi" class="tp3">
79      1. Lista de Confiança das Autoridades Certificadoras
80      credenciadas junto à ICP-Brasil
81      </p>
82      </a>
83      <a id="link"
84      href="https://homol-validar.iti.gov.br/internacional/tsl_nb.xml"
85      target="_blank" download>
86      <p id="listaConfi" class="tp4">
```

**Make sure not using rel="noopener" is safe here.**

Figura 30 – Nono ponto crítico de segurança encontrados no **middleware**

### 6.3.3 Análise dos dados do *front-end* do Validar

O *SonarQube* identificou 26 *hotspots* no **front-end** do objeto de análise deste estudo. Desses, 10 pontos críticos requerem revisão média e 16 requerem revisão baixa, conforme demonstrado na Figura 31.

Os pontos críticos de segurança com necessidade média de revisão são:

- **Injeção de código (RCE):** três pontos de acesso indicam que é necessário “Certificar-se de que o código *'javascript:'* seja seguro, pois é uma forma de *eval()*.”
- **Negação de serviço (DoS):** seis *hotspots* requerem “Certificar-se de que o *regex* usado aqui, vulnerável ao tempo de execução super linear devido ao retrocesso, não possa levar à negação de serviço.”
- **Permissão:** A imagem *httpd* está sendo executada como *root* por padrão. É necessário certificar-se de que essa configuração é segura.

Os pontos críticos de segurança com necessidade baixa de revisão são:



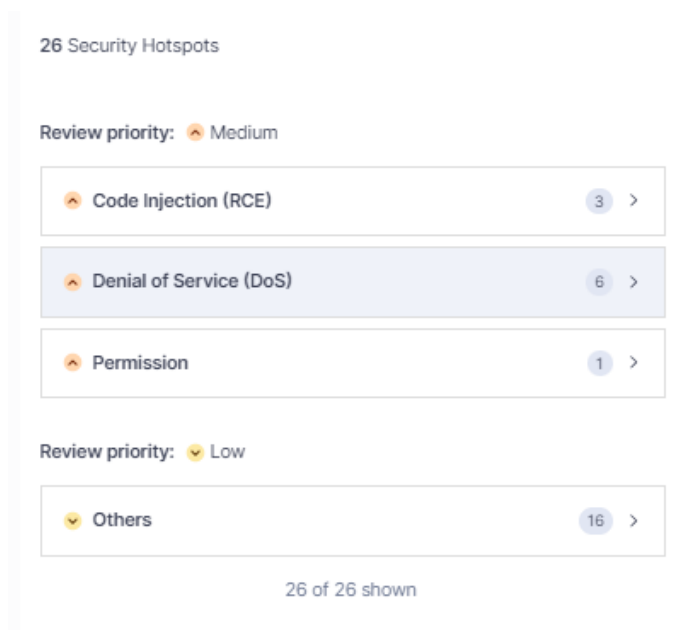


Figura 31 – Classificação de severidade dos pontos críticos de segurança encontrados no **front-end**.

- **Outros:**

- Nove pontos do tipo “Certificar-se de que não usar o recurso de integridade de recursos seja seguro aqui.”
- Sete pontos de título “Certificar-se de que não usar *rel=‘noopener’* seja seguro aqui.”

### 6.3.3.1 Pontos críticos da categoria de Injeção de código (RCE) encontrados *front-end*

Ao analisar os três pontos críticos de segurança do tipo injeção de código, foi constatado que todos se referem ao mesmo código e ao mesmo *hotspot*, representado pela Figura 32. Este ponto crítico de segurança possui um nível de severidade médio.

O uso de *javascript:* em links pode levar a vulnerabilidades de injeção de código, sendo explorado para injetar e executar código malicioso. Isso pode resultar em vulnerabilidades de injeção de código remoto (RCE), onde um atacante pode injetar e executar código arbitrário, e facilitar ataques de *Cross-Site Scripting (XSS)*, onde *scripts* maliciosos são executados no navegador de outros usuários. *Scripts* injetados podem manipular o DOM, roubar informações sensíveis ou redirecionar usuários para sites maliciosos (BONI, 2018).

A severidade é considerada média porque, embora a execução dinâmica de código represente um risco significativo, esse risco pode ser mitigado com a implementação de boas práticas de segurança. É importante ressaltar que o arquivo *core-init.js*, que apresenta o *hotspot*, é uma biblioteca externa que foi trazida para o projeto.

Make sure that 'javascript:' code is safe as it is a form of eval(). [🔗](#)

Dynamically executing code is security-sensitive [javascript:S1523](#)

**Status: To review**  
This security hotspot needs to be reviewed to assess whether the code poses a risk. [Review](#)

**Review priority:**  
🔴 Medium

**Category:**  
Code Injection (RCE)

**Assignee:**  
Not assigned [▼](#)

Where is the risk? What's the risk? Assess the risk How can I fix it? Activity

front/files/js/ref/core-init.js [📄](#) [Open in IDE](#)

```
8519     }  
8520   }  
8521 }  
8522 }  
8523  
8524   _createIntervalElement(type) {  
8525     const interval = document.createElement('li')  
8526     interval.setAttribute(`data-${type}-interval`, '')  
8527  
8528     const a = document.createElement('a')  
8529     a.setAttribute('href', 'javascript:void(0)')  
  
8530  
8531     const icon = document.createElement('i')  
8532     icon.classList.add('fas', 'fa-ellipsis-h')
```

Make sure that 'javascript:' code is safe as it is a form of eval().

Figura 32 – Pontos críticos da categoria de Injeção de código (RCE) encontrados *front-end*

### 6.3.3.2 Pontos de acesso da categoria de Negação de serviço (DoS) encontrados *front-end*

Foram encontrados seis *hotspots* da categoria de Negação de serviço, todos causados pelo uso de *regex*. O primeiro foi encontrado no arquivo *pdfCreator.js* (Figura 33) e os demais foram identificados no arquivo *requestHandler.js* (Figura 33).

Os pontos de acesso de segurança identificados possuem um nível de severidade médio devido ao risco de *backtracking* excessivo em *regexes*, que pode ser explorado para causar negação de serviço (DoS). Para mitigar esses riscos, é crucial revisar e otimizar as expressões regulares usadas, evitando padrões que possam causar *backtracking* excessivo (STAICU; PRADEL; DARMSTADT, 2018) e implementando boas práticas de segurança, como limites de tempo e uso de bibliotecas de *regexes* seguras (SAXENA, 2024). Implementar essas mudanças ajudará a garantir a eficiência e segurança da aplicação.

### 6.3.3.3 Ponto crítico de segurança sobre permissão encontrado *front-end*

Este *hotspot*, Figura 35, possui o mesmo comportamento e conseqüentemente a mesma análise do encontrado no item 6.3.2.1.

Make sure the regex used here, which is vulnerable to super-linear runtime due to backtracking, cannot lead to denial of service. [🔗](#)

Using slow regular expressions is security-sensitive [javascript:S5852](#)

Status: To review  
This security hotspot needs to be reviewed to assess whether the code poses a risk. [Review](#)

Review priority: 🔥 Medium

Category: Denial of Service (DoS)

Assignee: Not assigned ▼

Where is the risk? What's the risk? Assess the risk How can I fix it? Activity

front/files/js/bibliotecas/pdfCreator.js [🔗](#) [Open in IDE](#)

```
3     var validEmail = false;
4     if(emailRegex.test(email)){
5         validEmail = true;
6     }
7     return validEmail;
8 }
9
10
11
12 function confirmUrlRegex(url){
13     var urlRegex = /((h?t?t?p?s?:?\/?\/?)((a-z|[A-Z]).?_?-?)+(\.[a-z]+)+)/;
```

Make sure the regex used here, which is vulnerable to super-linear runtime due to backtracking, cannot lead to denial of service.

Figura 33 – Pontos de acesso da categoria de Injeção de código (RCE) encontrados *front-end*

### 6.3.4 Pontos críticos de segurança do tipo Outros

Dos 16 pontos críticos de segurança, 9 são *hotspots* relacionados à verificação de integridade de recursos, especificamente “Certificar-se de que não usar o recurso de integridade de recursos seja seguro aqui”. Todos esses pontos estão associados à importação de *scripts* de fontes externas, conforme ilustrado na Figura 36.

Os outros 6 pontos críticos de segurança encontrados no *front* são sobre a ausência do parâmetro *noopener*, como o próprio título dos *hotspots* acusam “Certificar-se de que não usar *rel='noopener'* seja seguro aqui”.

Como demonstrado na Figura 37, o trecho de código em questão utiliza um *switch* para abrir diferentes *URLs* em função de um id específico. Cada caso dentro do *switch* abre uma *URL* em uma nova janela ou aba do navegador, utilizando a função **window.open**. A ausência do parâmetro *noopener* ao usar *window.open* pode introduzir uma vulnerabilidade de segurança conhecida como “*tabnabbing*”. Essa vulnerabilidade permite que a página recém-aberta acesse a *window.opener*, que é uma referência à janela ou aba original que abriu a nova janela (OWASP, b). Isso pode ser explorado por atacantes para

**Make sure the regex used here, which is vulnerable to super-linear runtime due to backtracking, cannot lead to denial of service.** [↗](#)

Using slow regular expressions is security-sensitive [javascript:S5852](#)

**Status: To review**  
This security hotspot needs to be reviewed to assess whether the code poses a risk. Review

**Review priority:** ⬆ Medium

**Category:** Denial of Service (DoS)

**Assignee:** Not assigned ▼

Where is the risk?
What's the risk?
Assess the risk
How can I fix it?
Activity

front/files/js/services/requestHandler.js Open in IDE

```

1  validarArquivo = window.location.href.replace(/([a-z|A-Z]+\.\html)$/, '') + 'arquivo';
2  validarURL = window.location.href.replace(/([a-z|A-Z]+\.\html)$/, '') + 'url'
3  relSimples = window.location.href.replace(/([a-z|A-Z]+\.\html)$/, '') + 'simples'
4  relConformidade = window.location.href.replace(/([a-z|A-Z]+\.\html)$/, '') + 'conformidade'
5  downloadPdf = window.location.href.replace(/([a-z|A-Z]+\.\html)$/, '') + 'downloadPdf'

```

**Make sure the regex used here, which is vulnerable to super-linear runtime due to backtracking, cannot lead to denial of service.**

Figura 34 – Pontos críticos da categoria de Injeção de código (RCE) encontrados *front-end*

**The httpd image runs with root as the default user. Make sure it is safe here.** [↗](#)

Running containers as a privileged user is security-sensitive [docker:S6471](#)

**Status: To review**  
This security hotspot needs to be reviewed to assess whether the code poses a risk. Review

**Review priority:** ⬆ Medium

**Category:** Permission

**Assignee:** Not assigned ▼

Where is the risk?
What's the risk?
Assess the risk
How can I fix it?
Activity

front/Dockerfile Open in IDE

```

1  FROM httpd:2.4

```

**The httpd image runs with root as the default user. Make sure it is safe here.**

Figura 35 – Ponto de acesso de segurança sobre permissão encontrado *front-end*

```
17
18
19 <!-- Importar JS -->
20 <script type="text/javascript" src="./js/bibliotecas/jquery.min.js"></script>
21 <script type="text/javascript" src="./js/bibliotecas/jquery.mask.js"></script>
22 <script type="text/javascript" src="./js/bibliotecas/core-init.js"></script>
23 <script src="./js/bibliotecas/sweetalert2.all.min.js"></script>
24
25 <script src="./js/createmask.js"></script>
26
27 <!--<script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/1.3.5/jspdf.min.js"></script-->
28 <script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/1.5.3/jspdf.min.js"></script>
29
30
31 <script src="https://cdnjs.cloudflare.com/ajax/libs/html2canvas/0.4.1/html2canvas.min.js"></script>
32 <script src = "js/bibliotecas/pdfCreator.js"></script>
33
34 <script src="./js/bibliotecas/boostrap.js"></script>
35 <script src="./js/componentes/custon.js"></script>
36 <script type="text/javascript" src="./js/componentes/tarja.js"></script>
37 </head>
38
39 <body>
40 <script src="js/componentes/..."></script>
```

Make sure not using resource integrity feature is safe here.

Figura 36 – Pontos críticos de segurança relacionados à verificação de integridade de recursos encontrados no *front-end*

```
38
39
40 switch (id) {
41   case 'twitter':
42     window.open('https://twitter.com/itigovbr'); break
43   case 'youtube':
44     window.open('https://www.youtube.com/user/itidigital'); break
45   case 'face':
46     window.open('https://www.facebook.com/itigovbr'); break
47   case 'linkedin':
48     window.open('https://br.linkedin.com/company/itigovbr'); break
49   case 'insta':
50     window.open('https://instagram.com/itigovbr?igshid=YmMyMTA2M2Y='); break
51 }
52
53 function modalTermos() {
```

Make sure not using "noopener" is safe here.

Figura 37 – Pontos críticos de segurança relacionados à ausência do parâmetro *noopener* encontrados no *front-end*

redirecionar a página original ou executar *scripts* maliciosos, como já discutido mais acima no item 6.3.2.5.

## 7 Resultados do Teste de Segurança Dinâmico

Este capítulo descreve, em detalhes, o passo a passo realizado neste estudo de caso para coletar os dados do teste dinâmico de segurança (DAST) utilizando o OWASP Zap. Além disso, os dados coletados serão minuciosamente registrados, analisados e examinados nesta seção. Para tal finalidade, foram empregadas tabelas e gráficos.

### 7.1 Passo-a-passo da coleta de dados

O teste dinâmico de segurança foi executado por meio do *OWASP Zed Attack Proxy*, mais conhecido com *ZAP*. Para realizar o *download* desta ferramenta gratuita de DAST deve-se acessar o seu site<sup>1</sup> e selecionar a arquitetura da sua máquina. Ao concluir o *download* basta clicar no arquivo baixado para iniciar a instalação. A instalação no *ZAP* para o presente trabalho foi seguida o passo-a-passo recomendado pela aplicação.

Para iniciar o teste dinâmico de segurança do Validar nesta ferramenta, utilizou-se a opção de ataque automático. O ataque foi realizado após informar a URL do serviço<sup>2</sup>. Foram empregados tanto o *spider* tradicional quanto o *ajax spider*, configurado para navegadores com o *Chrome*. Após a configuração dessas opções, o ataque foi iniciado, e os resultados, juntamente com seus detalhes, foram exibidos na aba de alertas.

### 7.2 Dados coletados do teste de segurança dinâmico

Durante o teste de segurança dinâmica do serviço de assinaturas digitais, foram identificados 16 pontos de atenção, como ilustrado na figura 38. Entre eles, detectou-se uma vulnerabilidade de alto risco, além de cinco tipos de vulnerabilidades como de risco médio, resultando em 87 ocorrências de segurança nesse nível. Também foram encontrados seis tipos de vulnerabilidades de baixo risco, totalizando 305 ocorrências. Além disso, foram identificados cinco tipos de alertas informativos, somando 123 ocorrências de segurança.

O total de pontos de acesso de segurança identificados é de 516, sendo que, sem considerar os alertas informacionais, o número reduz-se a 393. É importante destacar que essas vulnerabilidades ainda passarão por uma análise mais aprofundada, e é possível que alguns dos alertas representam falsos positivos.

---

<sup>1</sup> <https://www.zaproxy.org/download/>

<sup>2</sup> <https://validar.iti.gov.br/>

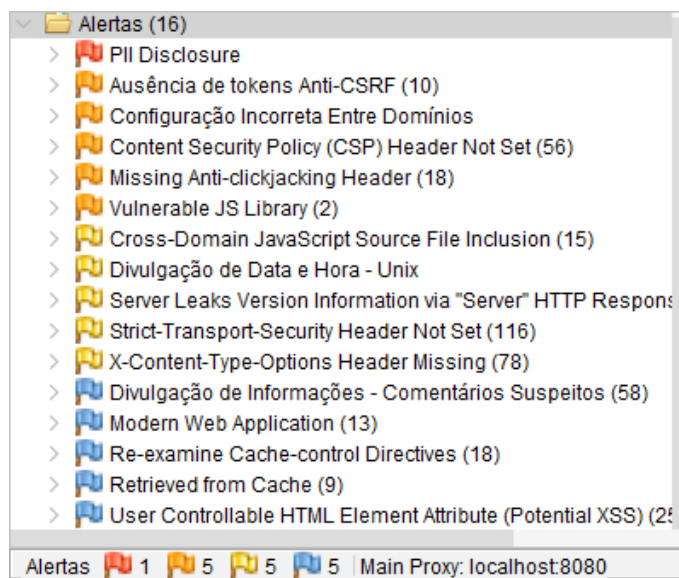


Figura 38 – Alertas identificados no DAST

### 7.2.1 Alertas de alto risco

Ao analisar a vulnerabilidade de alto risco de nome “*PII Disclosure*” (Divulgação de Informações Pessoais Identificáveis), Figura 39 na página ‘<https://validar.iti.gov.br/guia-desenvolvedor.html>’ foi constatado que esta é na verdade um falso positivo. Essa página é exclusivamente informativa, projetada para fornecer diretrizes aos desenvolvedores, sem permitir que os usuários insiram ou recebam quaisquer informações pessoais. Não há campos de formulário, autenticação ou qualquer outra funcionalidade que permita a coleta ou exibição de PII, tornando a possibilidade de divulgação de tais informações inexistentes.

A evidência apresentada, como o número ‘36705 55651 8985’, foi equivocadamente identificada como um número de cartão de crédito *DinersClub*. No entanto, essa sequência não representa dados reais ou sensíveis inseridos por usuários. A natureza estática e controlada do conteúdo da página elimina qualquer risco de exposição de dados privados.

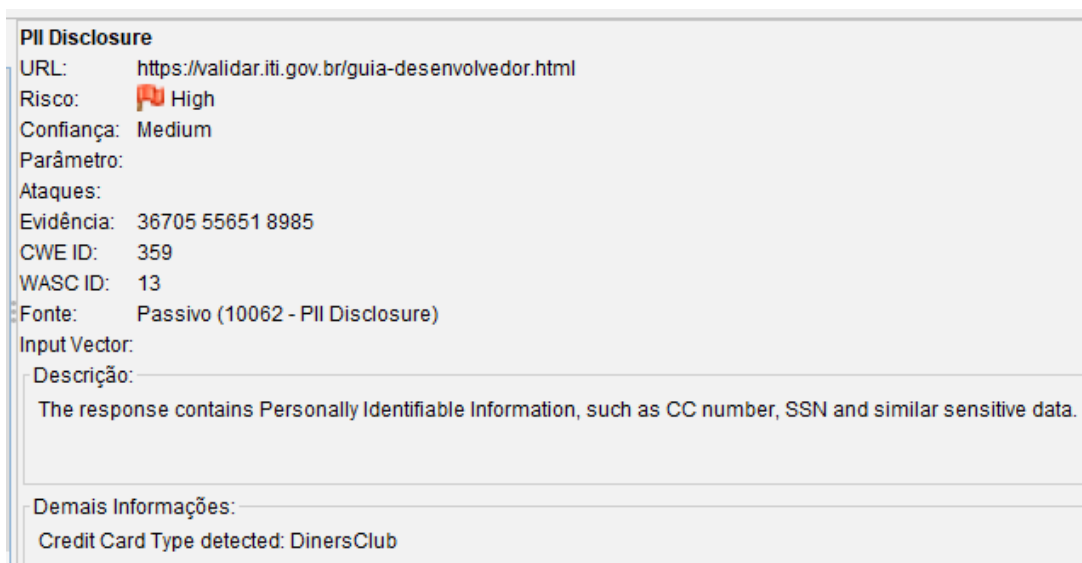


Figura 39 – Vulnerabilidade de alto risco identificada pelo ZAP

## 7.2.2 Alertas de risco médio

Ao realizar o teste de segurança dinâmico, os alertas de médio risco e suas quantidades retornadas foram:

- Ausência de tokens Anti-CSRF (10);
- Configuração Incorreta Entre Domínios;
- *Content Security Policy (CSP) Header Not Set* (56);
- *Missing Anti-clickjacking Header* (18);
- *Vulnerable JS Library*(2).

### 7.2.2.1 Ausência de tokens Anti-CSRF

O ponto de acesso de segurança do tipo **Cross-Site Request Forgery** foi mais uma vez identificado, agora no DAST. Isto ocorreu nesta modalidade de teste pois não foram localizados *tokens Anti-CSRF* no formulário de submissão HTML, como mostra a Figura 40. Como mencionado por Júnior (2024), isto possibilita falsificação de solicitação entre sites (CSRF), este tipo de ataque explora a confiança que um site tem em um usuário, e podem se tornar bem-sucedidas se:

- A vítima tem uma sessão ativa no site de destino;
- A vítima está autenticada por meio de autenticação HTTP no site de destino;
- A vítima está na mesma rede local do site de destino.



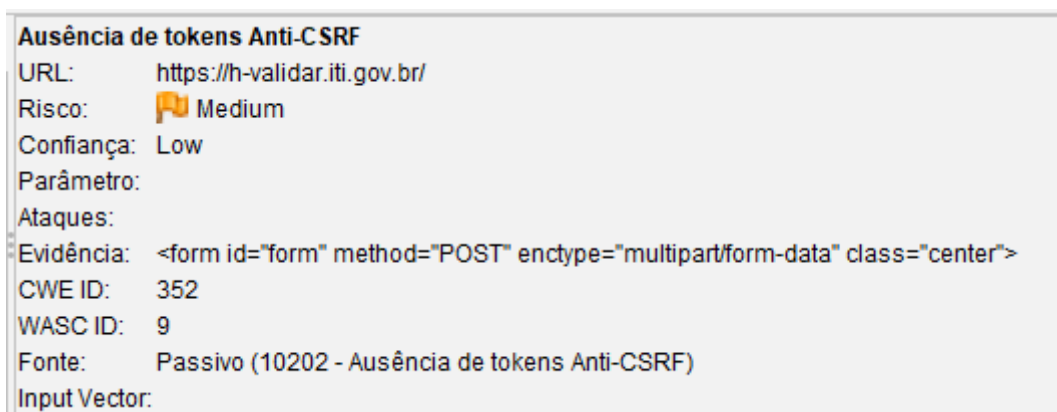


Figura 40 – Pontos críticos de segurança relacionados à ausência

A própria ferramenta ZAP trouxe as seguintes informações sobre este alerta:

“O CSRF era usado principalmente para executar ações contra um site-alvo usando os privilégios da vítima, mas técnicas recentes foram descobertas para vazamento de informações obtendo acesso às respostas. O risco de vazamento/divulgação não autorizada de informações aumenta drasticamente quando o site de destino é vulnerável a XSS, porque o XSS pode ser usado como uma plataforma para CSRF, permitindo que o ataque opere dentro dos limites da política de mesma origem”.

No caso do VALIDAR não é necessário que o usuário faça autenticação, a rede que o serviço está hospedado possui *firewall*, o que dificulta um ataque deste tipo seja bem sucedido, o que torna este alerta de baixo risco.

#### 7.2.2.2 Configuração Incorreta Entre Domínios

O DAST mais uma vez confirma a existência de uma possível vulnerabilidade levantada no SAST. Desta vez foi a possibilidade de carregamento de dados do navegador web, devido a uma configuração incorreta do *Cross Origin Resource Sharing* (CORS) no servidor web, como no item [6.3.2.4](#).

Mesmo com o Validar requisitando o uso de API *keys* para o uso da API é importante garantir a segurança dessas API *keys*, revisar e restringir a configuração CORS, para mitigar esse risco.

#### 7.2.2.3 Ausência do cabeçalho de *Content Security Policy* (CSP)

O OWASP ZAP alerta sobre a ausência do cabeçalho de *Content Security Policy* (CSP), indicando que o site pode estar vulnerável a certos tipos de ataques, como *Cross-Site Scripting* (XSS) e injeção de dados.

A falta de CSP pode permitir que *scripts* maliciosos sejam injetados, potencialmente comprometendo informações ou desconfigurando o site para distribuição de *malware* (SANTOS, 2023). Implementar uma política CSP é essencial para restringir o carregamento de conteúdo de fontes não autorizadas, protegendo assim o site contra execuções não autorizadas de *scripts* e mantendo a integridade dos dados, especialmente em contextos onde esses dados não devem ser alterados.

#### 7.2.2.4 Falta de um cabeçalho de proteção contra *clickjacking*

A ferramenta de DAST emitiu um alerta para a falta de um cabeçalho de proteção contra *clickjacking*. *Clickjacking*, o sequestro ou o roubo de cliques, é uma técnica de ataque onde o atacante engana o usuário a clicar em algo diferente do que ele percebe, estes cliques são utilizados para executar operações maliciosas (JÚNIOR, 2011). Para um serviço de validação de assinaturas eletrônicas, como o Validar, é essencial garantir a integridade e a segurança das operações realizadas. *Clickjacking* poderia comprometer a experiência do usuário e a segurança das operações, como a validação de documentos ou induzindo o usuário a realizar ações não intencionais.

#### 7.2.2.5 Biblioteca JS vulnerável

O *OWASP ZAP* identificou uma vulnerabilidade na biblioteca *Bootstrap* utilizada pelo site, especificamente na versão 4.1.3. Segundo Tenable (2019) as versões entre 3.X e 4.3.1, podem ser afetadas por uma vulnerabilidade de *Cross-Site Scripting (XSS)*. A recomendação é atualizar para a versão mais recente disponível do *Bootstrap* ou pelo menos superior a 4.3.1, que inclui correções de segurança e melhorias. Atualizar a versão é fundamental para mitigar riscos associados a vulnerabilidades conhecidas e para garantir a segurança e estabilidade do site.

### 7.2.3 Alertas de risco baixo

Durante o DAST, foram encontrados cinco tipos de alertas diferentes, sendo estes:

- *Cross-Domain JavaScript Source File Inclusion (15)*
- Divulgação de Data e Hora - Unix
- *Server Leaks Version Information via "Server" HTTP Response Header Field (95)*
- *Strict-Transport-Security Header Not Set (116)*
- *X-Content-Type-Options Header Missing (78)*

### 7.2.3.1 Inclusão de arquivo fonte *JavaScript* entre domínios

O alerta “*Cross-Domain JavaScript Source File Inclusion*” ou Inclusão de arquivo fonte *JavaScript* entre domínios, indica que o site está carregando um ou mais arquivos de *script JavaScript* de domínios de terceiros.

A inclusão direta de recursos *JavaScript* de origem cruzada no lado do cliente é uma prática amplamente adotada para consumir serviços de terceiros e utilizar bibliotecas externas em aplicativos web. No entanto, essa prática apresenta sérios riscos à segurança, pois o código externo é executado no mesmo contexto e com os mesmos privilégios que o código primário. Isso significa que qualquer vulnerabilidade ou falha de segurança presente no código de terceiros pode afetar diretamente a aplicação que o inclui. Se o conteúdo desses *scripts* for alterado ou o domínio de onde eles são servidos for comprometido, isso pode resultar na execução de código malicioso ou comprometer a segurança da aplicação, permitindo ações não intencionais ou maliciosas, como a manipulação de dados ou o roubo de informações sensíveis (MUSCH et al., 2019).

É crucial garantir que todos os *scripts* sejam carregados de fontes seguras e que o controle sobre essas fontes seja mantido pelo ITI. Implementar medidas de segurança, como CSP, e revisar periodicamente as fontes de *scripts*, são passos importantes para mitigar os riscos associados a essa prática.

### 7.2.3.2 Divulgação de Data e Hora - Unix

A notificação sobre a “Divulgação de Data e Hora - Unix” indica que a aplicação ou servidor web está expondo carimbos de data e hora em respostas HTTP ou outras saídas. Embora esses dados possam não parecer sensíveis à primeira vista, em certos contextos, podem representar riscos de segurança. No entanto, no contexto do VALIDAR, a divulgação de data e hora é essencial e intencional. Ela serve para registrar o momento exato em que a validação foi realizada, comprovando que o documento e suas assinaturas estavam íntegros e válidos naquele instante. Isso é fundamental para a confiança e a validade legal do processo de validação. O que faz dessa desse alerta um falso positivo.

### 7.2.3.3 Servidor vaza informações de versão por meio do campo de cabeçalho de resposta HTTP “Servidor”

O alerta “*Server Leaks Version Information via 'Server' HTTP Response Header Field*” indica que o servidor web ou de aplicação está divulgando informações sobre a versão do software utilizado, através do cabeçalho “*Server*” nas respostas HTTP. O OWASP ZAP alerta que esta prática pode ser arriscada, pois fornece a potenciais atacantes detalhes específicos sobre o ambiente do servidor, que podem ser usados para explorar vulnerabilidades conhecidas associadas a essa versão específica de software.

Entretanto, a divulgação da versão do serviço pelo ITI é intencional, permitindo que o usuário saiba exatamente qual versão foi utilizada para validar as assinaturas em seu documento. Isso é fundamental porque, com cada atualização, as regras de validação podem ser alteradas, o que pode levar a diferentes resultados na validação das assinaturas. Cada atualização visa aprimorar o serviço e torná-lo mais preciso. Portanto, a divulgação da versão é essencial para garantir a rastreabilidade e a transparência no processo de validação.

Isto torna a vulnerabilidade identificada um falso positivo, pois a prática é deliberada e alinhada com os objetivos de transparência e rastreabilidade do ITI, o que justifica sua manutenção. Contudo, outras medidas de segurança adequadas sejam implementadas para mitigar os riscos associados.

#### 7.2.3.4 *Strict-Transport-Security Header Not Set*

A notificação sobre a ausência do cabeçalho “*Strict-Transport-Security*” (HSTS) é a com maior número de aparições, alcançando o valor de 116 repetições. Esta vulnerabilidade ocorre quando o servidor web não está configurado para forçar o uso de conexões HTTPS. O HSTS é um mecanismo de segurança que assegura que os navegadores interajam com o servidor apenas por meio de conexões seguras (HTTPS), protegendo de ataques de interceptação (BATISTA et al., 2016). Uma vulnerabilidade semelhante foi identificada no SAST no item 6.3.2.3.

No contexto de um serviço de validação de assinaturas eletrônicas, a ausência do cabeçalho “*Strict-Transport-Security*” (HSTS) pode expor o sistema a riscos significativos. Sem HSTS, há um risco aumentado de ataques de interceptação (*Man-in-the-Middle*), onde um atacante pode capturar e manipular o tráfego entre o cliente e o servidor (TRENDMICRO, 2023).

#### 7.2.3.5 *X-Content-Type-Options Header Missing*

O retorno “*X-Content-Type-Options Header Missing*”, com 78 ocorrências, indica que o servidor web não está utilizando o cabeçalho “*X-Content-Type-Options*” com o valor “*nosniff*”. Esse cabeçalho é importante para prevenir que navegadores antigos, como versões mais antigas do *Internet Explorer* e *Chrome*, realizem a detecção MIME diretamente no corpo da resposta, o que pode levar a uma interpretação incorreta e exibição do conteúdo como um tipo de arquivo diferente do declarado originalmente (JÚNIOR, 2024). Isso representa um risco significativo, pois pode abrir brechas para ataques, incluindo a execução de *scripts* maliciosos.

## 7.2.4 Alertas de caráter informacional

O OWASP ZAP além de devolver possíveis vulnerabilidades de alto, médio e baixo risco também traz alertas informacionais. Nesta categoria foram encontrados cinco categorias, sendo estas:

- Divulgação de Informações - Comentários Suspeitos (58)
- *Modern Web Application* (13)
- *Re-examine Cache-control Directives* (18)
- *Retrieved from Cache* (9)
- *User Controllable HTML Element Attribute (Potential XSS)*(25)

### 7.2.4.1 Divulgação de Informações - Comentários Suspeitos

O alerta sobre “Divulgação de Informações - Comentários Suspeitos” indica que a resposta do servidor contém comentários que podem ser considerados suspeitos ou que revelam informações sensíveis. A ferramenta ZAP alerta que esses comentários podem fornecer pistas a potenciais atacantes sobre o funcionamento interno do sistema, vulnerabilidades ou até mesmo detalhes sensíveis como nomes de usuários.

A presença de comentários suspeitos ou que contenham informações sensíveis no código representa um risco de segurança significativo. Remover esses comentários e corrigir qualquer problema subjacente é essencial para proteger a integridade e a confidencialidade do sistema, além de alinhar-se com as melhores práticas de segurança cibernética.

### 7.2.4.2 *Modern Web Application*

O retorno “*Modern Web Application*” indica que o aplicativo parece ser um aplicativo web moderno. A própria ferramenta deixa claro que não se trata de uma funcionalidade, é apenas um alerta informativo e, portanto, nenhuma alteração é necessária. Além disso, traz a informação “Foram encontrados links com o destino “*self*” - isso é frequentemente usado por estruturas modernas para forçar o recarregamento completo da página.” para evidenciar que se trata de uma aplicação moderna.

### 7.2.4.3 *Re-examine Cache-control Directives*

A notificação sobre as diretivas de controle de cache destaca a importância de uma configuração adequada dos cabeçalhos de controle de cache para evitar o armazenamento não autorizado de conteúdo sensível, comprometendo a confidencialidade e a integridade dos dados. A ausência ou má configuração dessas diretivas pode permitir que navegadores

e *proxies* acessem e alterem o conteúdo em cache, abrindo espaço para vulnerabilidades como o envenenamento de cache, onde um atacante pode manipular, por exemplo, o Sistema de Nomes de Domínio (DNS) (FRANZESE, 2023).

No entanto, no serviço Validar, o cache não armazena informações sensíveis. Os dados de navegação coletados são anônimos e utilizados apenas para registrar as configurações e preferências dos usuários e gerar relatórios estatísticos. Essa prática está claramente delineada no termo de uso do serviço, sob o tópico TRATAMENTO DE DADOS, garantindo que a privacidade e a segurança dos usuários sejam mantidas, como demonstrada o seguinte recorte deste:

“Durante a navegação, o portal VALIDAR utiliza *cookies* próprios (primários), ou seja, dos domínios validar.iti.gov.br e validar.iti.br, para registrar as configurações e preferências de navegação dos usuários e gerar relatórios estatísticos através do *Google Analytics*, e também *cookies* de terceiros para complementar essas estatísticas. A configuração do *Google Analytics* para os domínios citados não permite que dados pessoais e o documento assinado objeto de validação sejam armazenados e/ou repassados a terceiros.”

#### 7.2.4.4 *User Controllable HTML Element Attribute (Potential XSS)*

A notificação '*User Controllable HTML Element Attribute (Potential XSS)*' indica uma possível vulnerabilidade de *Cross-Site Scripting (XSS)* na aplicação, onde entradas fornecidas pelo usuário através de parâmetros de sequência de consulta e dados POST podem controlar diretamente os valores de atributos HTML. Essa falha pode permitir que um invasor injete *scripts* maliciosos, resultando na execução de código não autorizado no contexto do navegador do usuário. Por isso, é essencial validar e sanitizar todas as entradas do usuário para prevenir a execução de *scripts* maliciosos. Estas informações podem ser encontradas no próprio alerta do ZAP<sup>3</sup>.

<sup>3</sup> <https://www.zaproxy.org/docs/alerts/10031/>

## 8 Análise Conjunta dos Resultados das Três Técnicas

Este capítulo é dedicado à análise, comparação e discussão dos resultados obtidos a partir das três técnicas avaliadas: modelagem de ameaças, SAST e DAST. Além disso, serão apresentadas as sugestões de mitigação para as vulnerabilidades identificadas. Para esta análise não serão considerados os falsos positivos identificados.

### 8.1 Dados coletados

Para facilitar a compreensão e a análise das possíveis vulnerabilidades encontradas, estas foram agrupadas conforme o tipo de problema ao qual estão relacionadas. Por exemplo, todas as vulnerabilidades classificadas como “*Script* entre sites” foram consolidadas e representadas por uma única entrada com essa denominação. Dentre as vulnerabilidades que são do tipo “*Script* entre sites” estão que envolve *Cross-Site Scripting (XSS)* e *Cross-Site Request Forgery (CSRF)*. Algumas vulnerabilidades se encaixam em mais de uma categoria, por exemplo “Alteração do fluxo de execução” e “Importação de *script* externos”, e serão contabilizadas em cada uma.

ID	Vulnerabilidade	Modelagem de ameaça	SAST	DAST
01	<i>Script</i> entre sites	5	5	96
02	Elevação de privilégio	20	2	0
03	Memória adulterada	2	-	-
04	Potencial repúdio	8	-	-
05	Negação de serviço(DoS)	12	9	116
06	Falsificação da entidade de destino externo	8	7	116
07	Fluxo de dados interrompidos	8	9	116
08	Execução remota de código	4	9	113
09	Alteração do fluxo de execução	4	10	113
10	Divulgação de Informações	-	1	136
11	Importação de <i>script</i> externos	-	9	35

Tabela 6 – Análise conjunta dos resultados

Ao analisar a quantidade de pontos de acesso identificados por cada técnica e o nível de severidade associado a cada um, conforme a Figura 41, observa-se que a modelagem de ameaças foi a que mais identificou possíveis vulnerabilidades com alto nível

## Comparação entre as técnicas

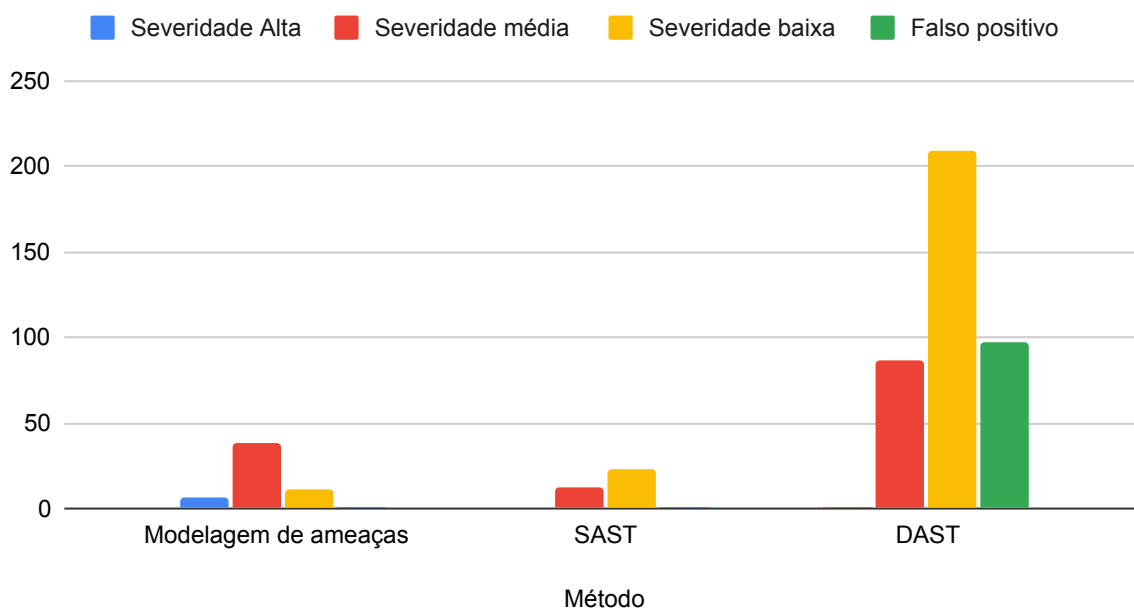


Figura 41 – Comparação dos resultados obtidos nos três métodos utilizados no estudo de caso

de criticidade, totalizando 6 ocorrências. O SAST e o DAST não identificaram nenhuma vulnerabilidade de alto nível de criticidade.

Em relação às vulnerabilidades de severidade média, a modelagem de ameaças identificou 38, o SAST encontrou 12, e o DAST detectou 87. Para vulnerabilidades de baixo nível de criticidade, a modelagem de ameaças identificou 11, o SAST encontrou 23, e o DAST detectou 209. Para este cálculo não foram considerados os falsos positivos.

Além disso, o DAST foi a técnica que identificou o maior número total de vulnerabilidades e também de falsos positivos confirmados, totalizando 97 ocorrências. O SAST apresentou apenas um falso positivo.

## 8.2 Análise conjunta dos dados

Ao observar a Tabela 6 e a Figura 41 nota-se algumas coisas que chamam a atenção, tais como o número superior de possíveis vulnerabilidades encontradas pelo teste dinâmico de segurança e a quantidade de falsos positivos apontados por este. Além disso, alguns tipos de vulnerabilidades foram identificadas apenas na modelagem de ameaças, enquanto que duas categorias não foram identificadas na modelagem, apenas pelo SAST e pelo DAST.



O DAST foi o método que identificou uma maior quantidade de vulnerabilidades, isso se deve ao fato que este teste é realizado no tempo de execução do software para assim verificar a funcionalidade que só é aparente quando todos os componentes estão integrados e funcionando (Laurie Williams, 2021). Outra razão para o elevado número de detecções no DAST é que essa técnica pode estar cobrindo áreas da aplicação que os outros métodos não conseguem analisar de forma tão eficaz, pois essas áreas são geradas dinamicamente e só se manifestam durante a execução da aplicação.

Mais um ponto que se destaca ao observar a Tabela 6 é que as possíveis vulnerabilidades das categorias de Elevação de Privilégio, Memória Adulterada e Potencial de Repúdio foram identificadas apenas na modelagem de ameaças, com poucas ou nenhuma ocorrência nas outras técnicas. Isso ocorre porque a modelagem de ameaças é uma análise da arquitetura do serviço que deve ser realizada antes da sua implementação. Dessa forma, é possível tratar os pontos de atenção identificados por esse método durante o processo de desenvolvimento, evitando que essas vulnerabilidades se concretizem. Provavelmente, foi isso que aconteceu neste caso; a forma como o serviço foi desenvolvido abordou e mitigou essas possíveis vulnerabilidades antecipadamente.

A ausência de identificação das vulnerabilidades de Divulgação de Informação e Importação de Scripts Externos na modelagem de ameaças é um ponto que merece atenção. Embora uma explicação possível para essas ocorrências seja a presença de falsos positivos nos outros métodos (SAST e DAST), essa possibilidade é improvável, já que ambas as técnicas identificaram essas vulnerabilidades de forma consistente.

O motivo mais provável para a modelagem de ameaças não ter previsto essas vulnerabilidades é que essa técnica se concentra em ameaças previsíveis baseadas na estrutura do sistema, mas pode não detectar problemas que dependem de implementações específicas ou configurações que só se tornam evidentes em fases posteriores. Vulnerabilidades como Divulgação de Informação e Importação de *Scripts* Externos geralmente dependem de detalhes de implementação ou de configurações em tempo de execução, que são difíceis de prever durante a fase de modelagem de ameaças.

Essas vulnerabilidades podem surgir devido a interações complexas entre diferentes componentes, permissões de segurança inadequadas ou outros fatores específicos que só são identificados ao analisar o código em execução (como no DAST) ou através de uma inspeção detalhada do código fonte (como no SAST). Portanto, a modelagem de ameaças, por ser uma análise teórica e de alto nível, pode não capturar todos os riscos que só se manifestam em condições de implementação específicas ou durante a operação do sistema.

Ao analisar os resultados das três técnicas, percebe-se a confirmação da presença de certos tipos de vulnerabilidades, como *Script* entre Sites, Negação de Serviço, Falsificação de Entidade Externa, Interrupção de Fluxo de Dados, Execução Remota de Código e Alteração do Fluxo de Execução. É importante destacar que a vulnerabilidade que apre-

sentou um número elevado de ocorrências em todos os métodos foi a Negação de Serviço.

Os resultados obtidos pela análise conjunta dos três métodos corrobora o que a literatura sobre o assunto afirma. A modelagem de ameaças é essencial para prever possíveis pontos de acesso ao sistema em desenvolvimento, de modo que nem permita que estes venham a existir. Os testes de segurança estático e dinâmico são complementares, pois juntos fornecem uma cobertura mais abrangente das vulnerabilidades de uma aplicação. Enquanto o SAST examina a base do código para detectar problemas na lógica e segurança da codificação, o DAST avalia o comportamento da aplicação em um ambiente em execução, detectando vulnerabilidades que surgem em tempo real. Em alguns casos, tanto o SAST quanto o DAST identificaram a mesma vulnerabilidade. Nesses casos, o tópico responsável pela análise da vulnerabilidade na DAST indicou qual era a vulnerabilidade correspondente identificada pelo SAST.

### 8.3 Propostas de mitigação

Esta seção tem como objetivo apresentar propostas de mitigação para as vulnerabilidades de alta severidade identificadas. No total, foram identificadas 6 vulnerabilidades com alto nível de criticidade, todos estes identificados na modelagem de ameaças.

É relevante destacar que as vulnerabilidades classificadas como de alta criticidade na modelagem de ameaças também foram identificadas nas análises de SAST e DAST, embora com um nível de severidade inferior. Ainda que apresentem menor criticidade nessas técnicas, serão propostas medidas para mitigá-las ou, ao menos, reduzir seu nível de severidade, de modo a não comprometer a segurança da aplicação. A adoção dessas ações é justificada pelo fato de que, na operação cotidiana da aplicação, essas vulnerabilidades representam os problemas mais recorrentes e críticos enfrentados pelo serviço.

Na modelagem de ameaças, foram identificadas as seguintes vulnerabilidades:

- Um item do tipo Falsificação da Entidade Externa do Usuário Humano, pertencente à Categoria de Falsificação, com o ID 34.
- Três itens do tipo Falha ou Parada Potencial do Processo para API, pertencentes à Categoria de Negação de Serviço, com os IDs 44, 54 e 64.
- Dois itens do tipo Interrupção Potencial do Fluxo de Dados HTTPS, também pertencentes à Categoria de Negação de Serviço, com os IDs 45 e 65.

Essas vulnerabilidades de alta severidade serão o foco das propostas de mitigação apresentadas a seguir. Se alguma não for passível de completa mitigação será apresentado com reduzir o nível de criticidade ao ponto que esta seja aceitável.

### 8.3.1 Falsificação da Entidade Externa do Usuário Humano

Para o item Falsificação da Entidade Externa do Usuário Humano, pertencente à Categoria de Falsificação, ID 34, a seguinte descrição é trazida pelo TMT: “O usuário humano pode ser falsificado por um invasor e isso pode levar ao acesso não autorizado ao *Proxy Apache*. Considere o uso de um mecanismo de autenticação padrão para identificar a entidade externa”. A forma mais provável de essa falsificação ocorrer é por meio de *bots*. Para evitar que isso aconteça, o VALIDAR pode incluir mecanismos de autenticação, como sugerido pelo TMT. Entretanto, outras opções incluem o uso de *CAPTCHA*, limitação de requisições (*Rate Limiting*) ou a implementação de autenticação multi-fator (MFA). É importante verificar qual estratégia seria a mais adequada para o serviço, de forma que a experiência do usuário não seja comprometida.

A falsificação da Entidade Externa do Usuário Humano também pode ocorrer de outras maneiras além do uso de *bots*, como ataques manuais por usuários mal-intencionados. Para mitigar esse risco, é fundamental implementar a validação das entradas provenientes de entidades externas, garantindo que os dados sejam verificados, sanitizados e validados para estarem no formato esperado e sem conteúdo malicioso. Essa prática ajuda a prevenir ataques como injeção de SQL e *Cross-Site Scripting* (XSS), que exploram falhas na manipulação de entrada, ainda que não estejam diretamente relacionados à falsificação de identidade (OWASP, a).

Outra sugestão é implementar o monitoramento contínuo e a auditoria detalhada das interações com o sistema, como a *Microsoft* faz, para detectar atividades suspeitas (GLUCKD, 2024). Assim serão registrados todas as solicitações e respostas, incluindo tentativas de acesso falhas, permite uma rápida identificação e resposta a tentativas de falsificação ou acesso não autorizado.

Além disso, garantir que todas as comunicações sejam feitas via HTTPS para proteger a integridade e confidencialidade dos dados. A implementação de cabeçalhos de segurança adicionais, como *HSTS* (*HTTP Strict Transport Security*), reforça o uso de conexões seguras e previne interceptações e ataques de *man-in-the-middle* (TRENDMICRO, 2023).

### 8.3.2 Falha ou parada potencial do processo para API

O item Falha ou parada potencial do processo para API, referente à categoria Negação de Serviço (DoS), aparece três vezes na modelagem de ameaças com os IDs 44, 54 e 64. O *Threat Modeling Tool* (TMT) traz a seguinte descrição para essas ameaças: “A API trava, interrompe, para ou é executada lentamente; em todos os casos, violando uma métrica de disponibilidade”. Esse comportamento ocorre principalmente devido a uma sobrecarga da API, que pode ser causada por diversos fatores. No contexto do VALIDAR,

com base na análise e na experiência com o serviço, os fatores mais prováveis de causar sobrecarga são:

- **Grande quantidade de requisições em um mesmo momento**, seja por meio de usuários legítimos, bem-intencionados ou não, e por *bots*, como mencionado no tópico anterior. A alta demanda simultânea pode saturar os recursos disponíveis;
- **Documentos com número elevado de assinaturas**, principalmente devido à má compreensão de que o conceito de rubricas não se aplica a documentos digitais. Isso pode gerar um consumo excessivo de recursos internos, impactando negativamente o desempenho da API;
- **Falta de escalabilidade do sistema**, seja ela vertical (aumento de recursos em um único servidor) ou horizontal (adição de mais servidores) (PACHECO, 2023), o que impede o sistema de lidar adequadamente com picos de demanda;
- **Dependências externas lentas ou falhas**, especialmente para a verificação de documentos com assinaturas internacionais e para a validação de registro de profissionais de saúde nos seus respectivos conselhos. Essas dependências podem introduzir atrasos significativos ou causar falhas na API;
- **Problemas de rede**, como latência elevada, perda de pacotes ou instabilidade, que podem causar falhas na comunicação entre a API e outros sistemas, comprometendo a disponibilidade e o desempenho do serviço.

No contexto do VALIDAR, problemas de grande quantidade de requisições simultâneas, tanto de usuários legítimos quanto de *bots*, podem ser mitigados por meio de definição de limitação de requisições - *Rate Limiting* (CLOUDFLARE, ), CAPTCHA, balanceamento de carga e escalabilidade automática - *auto-scaling* (MOORE, ). Essas medidas ajudam a distribuir o tráfego e evitar sobrecargas.

Documentos com um número elevado de assinaturas, devido à má compreensão sobre o conceito de rubricas digitais, podem ser gerenciados através da educação dos usuários, como o serviço já faz por meio do Guia de boas práticas <sup>1</sup>.

A falta de escalabilidade pode ser resolvida com *auto-scaling* horizontal, que permite a adição automática de mais servidores, e a otimização de recursos verticais (PACHECO, 2023). Dividir a API em microsserviços também facilita a escalabilidade de diferentes componentes conforme a demanda (MONTE, 2020). Para o Validar essa alteração na arquitetura seria muito interessante, pois como demonstrado no Capítulo 4 esta é formada por camadas e serviços, o que a torna uma ótima candidata ao uso de microsserviços.

---

<sup>1</sup> <https://validar.iti.gov.br/guia.html>

Dependências externas lentas, como verificações de assinaturas internacionais ou validações de registro, podem ser tratadas com *Circuit Breaker*, um mecanismo que interrompe temporariamente requisições a serviços externos quando falhas ou lentidão são detectadas, evitando sobrecarga (PIVETTA, 2023). Juntamente com o *timeout* já implementado, isso garante que problemas externos não afetem o desempenho da API principal.

Por fim, problemas de rede como latência e perda de pacotes podem ser mitigados com o uso de CDNs (*Content Delivery Networks*), que são redes de servidores distribuídos geograficamente para entregar conteúdo de forma mais rápida (VAKALI; PALLIS, 2003). Além disso, o monitoramento de rede, melhorias na infraestrutura e o uso de *proxies* distribuídos também contribuem para melhorar a estabilidade e a velocidade do serviço.

### 8.3.3 Interrupção Potencial do Fluxo de Dados HTTPS

A ameaça descrita como interrupção potencial do fluxo de dados HTTPS, do tipo Negação de Serviço (DoS), aparece duas vezes na modelagem de ameaças, com os IDs 45 e 65. Segundo a descrição apresentada pelo TMT, esta ameaça ocorre quando “um agente externo interrompe o fluxo de dados através de um limite de confiança em qualquer direção”. Isso pode comprometer a disponibilidade do serviço ao bloquear ou interferir no tráfego de dados, afetando tanto a entrada quanto a saída de informações entre o cliente e o servidor.

Por se tratar de uma ameaça relacionada à negação de serviço, algumas das medidas de mitigação propostas no tópico anterior, como *Rate Limiting* e o uso de CDNs, também são aplicáveis a essa vulnerabilidade. Além disso, as soluções apresentadas anteriormente, no tópico , como a utilização de HTTPS em conjunto com cabeçalhos HSTS, para que assim ataques de *man-in-the-middle* sejam prevenidos. A implementação combinada dessas estratégias contribuirá significativamente para a eliminação dessa vulnerabilidade.

## 9 Conclusão

O presente estudo de caso teve como objetivo realizar uma avaliação de segurança no VALIDAR, serviço de validação de assinaturas eletrônicas oferecido pelo Instituto Nacional de Tecnologia da Informação (ITI). Para isso, foram utilizados métodos usados nas fases do *Microsoft Security Development Lifecycle (SDL)*. As técnicas empregadas do SDL foram: (1) a modelagem de ameaças com o auxílio da ferramenta *Microsoft Threat Modeling Tool (TMT)*, juntamente com os *frameworks* STRIDE e o DREAD; (2) a análise estática de código (SAST) por meio da ferramenta *SonarQube*; e (3) a análise dinâmica de segurança (DAST) utilizando o OWASP ZAP.

Os dados coletados em cada fase foram primeiramente analisados individualmente e, posteriormente, integrados para uma visão mais abrangente. A análise identificou, principalmente, ameaças potenciais de ataques de negação de serviço (DoS). Cada vulnerabilidade encontrada foi analisada e priorizada com base em seu nível de criticidade. Para as vulnerabilidades de alta severidade, foram propostas soluções visando mitigar esses riscos, com o objetivo de aumentar a resiliência e a segurança da aplicação.

Com base nos resultados, retomamos as questões que nortearam o estudo, apresentadas na Seção 1. A primeira questão foi: “O VALIDAR possui falhas de segurança severas?”. A avaliação revelou que não há presença de vulnerabilidades graves propriamente ditas, uma vez que as vulnerabilidades de alta severidade foram identificadas apenas na modelagem de ameaças e não pelas técnicas que analisam o código e sua execução. Além disso, nenhuma dessas ameaças identificadas compromete diretamente a integridade dos resultados das validações realizadas pelo VALIDAR.

A segunda questão, “Como essas falhas impactam o serviço?”, foi respondida com base nos riscos identificados. Embora as falhas possam resultar em negação de serviço ou lentidão excessiva, elas não afetam a confiabilidade dos resultados apresentados pelo serviço.

A terceira questão, “Quais são os principais fatores que contribuem para essas vulnerabilidades?”, revelou que a ausência de mecanismos de autenticação robustos, como MFA ou CAPTCHA, contribui para a vulnerabilidade à falsificação da entidade externa do usuário humano. Além disso, problemas relacionados à sobrecarga de requisições, falta de escalabilidade, dependências externas lentas e falhas de rede afetam a disponibilidade do serviço. A interrupção do tráfego HTTPS também é agravada pela falta de medidas preventivas, como *rate limiting* e *HSTS*.

A quarta questão, “É possível melhorar o nível de segurança do serviço?”, foi respondida afirmativamente. Sim, é possível melhorar a segurança do VALIDAR.

Por fim, a quinta questão, “Quais ações são necessárias para torná-lo mais seguro?”, trouxe propostas de ações. Para mitigar a falsificação da entidade externa, é necessário implementar mecanismos como CAPTCHA, *rate limiting* e MFA. O monitoramento contínuo e a adoção de HTTPS com HSTS também são essenciais. Para resolver os problemas de sobrecarga da API, recomenda-se implementar *rate limiting*, balanceamento de carga e *auto-scaling*. As dependências externas podem ser gerenciadas com *circuit breaker* e *timeouts*, enquanto o uso de CDNs melhora a estabilidade do serviço.

No caso da interrupção do tráfego HTTPS, as medidas sugeridas incluem *rate limiting*, *CDNs* e *HSTS* para proteger a comunicação e evitar ataques de negação de serviço.

Algumas ações, como *rate limiting* e *HSTS*, foram mencionadas em múltiplos tópicos, evidenciando que são medidas prioritárias e eficazes para iniciar as melhorias de segurança no VALIDAR.

# Referências

- ALVES, J. V. A. Melhorias na Infraestrutura do Sistema de Cadastro de Atividades Docente. 2022. Citado na página 67.
- ARTEM, L. et al. ELIMINATING PRIVILEGE ESCALATION TO ROOT IN CONTAINERS RUNNING ON KUBERNETES. Citado na página 66.
- BARBOSA, A. A. Análise estática de vulnerabilidades em software sob múltiplas abordagens. mar. 2015. Disponível em: <<https://bdm.unb.br/handle/10483/11782>>. Citado 2 vezes nas páginas 29 e 31.
- BATISTA, R. R. et al. Teste de invasão no SIGAA da UFPB. *GESTÃO.Org*, v. 14, n. Extra 5, p. 247–254, 2016. ISSN 1679-1827. Publisher: Universidade Federal de Pernambuco (UFPE) Section: GESTÃO.Org. Disponível em: <<https://dialnet.unirioja.es/servlet/articulo?codigo=7353513>>. Citado na página 83.
- BEHRENS, F. A Assinatura Eletrônica como Requisito de Validade dos Negócios Jurídicos e a Inclusão Digital na Sociedade Brasileira. 2005. Citado na página 39.
- BONI, L. R. Injeção de código malicioso em aplicações Web. 2018. Citado na página 72.
- BRERETON, P. et al. Using a protocol template for case study planning. *Proceedings of EASE*, v. 2008, jan. 2008. Citado 4 vezes nas páginas 17, 33, 34 e 35.
- CALIL, F. M. Desenvolvimento de Software Seguro: propostas para a melhoria da formação dos Engenheiros de Software nas universidades brasileiras. 2023. Citado 4 vezes nas páginas 16, 21, 23 e 24.
- CLOUDFLARE. *O que é limitação de taxa? | Limitação de taxa e bots*. Disponível em: <<https://www.cloudflare.com/pt-br/learning/bots/what-is-rate-limiting/>>. Citado na página 91.
- DOWD, M.; MCDONALD, J.; SCHUH, J. *The Art of Software Security Assessment: Identifying and Preventing Software Vulnerabilities*. [S.l.]: Pearson Education, 2006. ISBN 978-0-13-270193-8. Citado na página 21.
- ELDER, S. et al. Do I really need all this work to find vulnerabilities?: An empirical case study comparing vulnerability detection techniques on a Java application. *Empirical Software Engineering*, v. 27, n. 6, p. 154, nov. 2022. ISSN 1382-3256, 1573-7616. Disponível em: <<https://link.springer.com/10.1007/s10664-022-10179-6>>. Citado na página 17.
- EXPLOITING CORS – How to Pentest Cross-Origin Resource Sharing Vulnerabilities. 2023. Disponível em: <<https://www.freecodecamp.org/news/exploiting-cors-guide-to-pentesting/>>. Citado na página 67.
- FRANZESE, L. F. Um Estudo das Vulnerabilidades do OWASP Top 10 na plataforma Moodle. 2023. Citado na página 85.



GATES. *Bill Gates: Trustworthy Computing* \textbar WIREd. 2002. Disponível em: <<https://www.wired.com/2002/01/bill-gates-trustworthy-computing/>>. Citado na página 22.

GLUCKD. *Audit logging and monitoring overview - Microsoft Service Assurance*. 2024. Disponível em: <<https://learn.microsoft.com/en-us/compliance/assurance/assurance-audit-logging>>. Citado na página 90.

HEWKO, A. *STRIDE Threat Modeling* \textbar What Is It? \textbar Explanation and Examples \textbar Read. 2021. Publication Title: Software Secured. Disponível em: <<https://www.softwaresecured.com/stride-threat-modeling/>>. Citado 3 vezes nas páginas 9, 27 e 28.

ITI. *CertificadoDeAtributo.pdf*. Disponível em: <[https://itipr.sharepoint.com/sites/ProjetoCertificadodeAtributo324/Documentos%20Compartilhados/Certificado-Atributo\\_ANVISA-controle-receitas%20\(2\).pdf](https://itipr.sharepoint.com/sites/ProjetoCertificadodeAtributo324/Documentos%20Compartilhados/Certificado-Atributo_ANVISA-controle-receitas%20(2).pdf)>. Citado 2 vezes nas páginas 7 e 40.

ITI. *O ITI*. Publication Title: Instituto Nacional de Tecnologia da Informação. Disponível em: <<https://www.gov.br/iti/pt-br/aceso-a-informacao/institucional/o-iti>>. Citado na página 38.

ITI. *Resoluções*. 2021. Publication Title: Instituto Nacional de Tecnologia da Informação. Disponível em: <<https://www.gov.br/iti/pt-br/assuntos/legislacao/resolucoes/resolucoes>>. Citado 5 vezes nas páginas 7, 40, 41, 44 e 45.

ITI. *Validar*. 2022. Disponível em: <<https://validar.iti.gov.br/informacoes.html>>. Citado 2 vezes nas páginas 41 e 44.

JASPER, N. *Desenvolvendo controles de segurança baseados em modelagem de ameaças e análise de requisitos*. Tese (PhD Thesis), jun. 2014. Citado 2 vezes nas páginas 23 e 25.

JUNIOR, A.; JUSTINO, A. *Crimes cibernéticos no Brasil: suas causas e consequências em decorrência do crescimento da Pandemia do COVID-19*. [S.l.], 2022. Disponível em: <<http://104.207.146.252:8080/xmlui/handle/123456789/224>>. Citado na página 16.

JÚNIOR, N. S. J. D. S. *Mitigando os Efeitos dos Ataques de Frame–Alternativas às Brechas Inevitáveis*. 2011. Citado na página 81.

JúNIOR, J. E. d. S. *Análise de Segurança Cibernética no Tribunal de Contas do Estado do Rio Grande do Norte: Aplicação dos Princípios OWASP na Identificação e Mitigação de Vulnerabilidades*. 2024. Citado 2 vezes nas páginas 79 e 83.

JúNIOR, S.; DOS, I. E. *Segurança de software*. dez. 2013. Disponível em: <<https://ric.cps.sp.gov.br/handle/123456789/1209>>. Citado 2 vezes nas páginas 20 e 21.

LADISA, P. et al. *The Hitchhiker’s Guide to Malicious Third-Party Dependencies*. In: *Proceedings of the 2023 Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses*. [s.n.], 2023. p. 65–74. ArXiv:2307.09087 [cs]. Disponível em: <<http://arxiv.org/abs/2307.09087>>. Citado na página 68.

Laurie Williams. Secure Software Lifecycle Knowledge Area Version . . v. 1.0.2, 2021. Citado 6 vezes nas páginas 16, 21, 22, 23, 25 e 88.

LIMA, I. et al. Avanço da telemedicina no Brasil no período de pandemia da COVID-19: uma revisão sistemática da literatura / Advancement of telemedicine in Brazil during the COVID-19 pandemic period: a systematic review of the literature. *Brazilian Journal of Health Review*, v. 5, p. 10505–10525, maio 2022. Citado na página 15.

MAGALHÃES, R.; VENDRAMINI, A. Os impactos da quarta revolução industrial. *GV-executivo*, v. 17, n. 1, p. 40, mar. 2018. ISSN 1806-8979. Disponível em: <<http://bibliotecadigital.fgv.br/ojs/index.php/gvexecutivo/article/view/74093>>. Citado na página 15.

MARIJAN, B. *Swap Space in Linux: What It Is & How It Works*. 2023. Disponível em: <<https://phoenixnap.com/kb/swap-space>>. Citado na página 65.

Microsoft. *Ameaças - Microsoft Threat Modeling Tool - Azure*. 2023. Disponível em: <<https://learn.microsoft.com/pt-br/azure/security/develop/threat-modeling-tool-threats>>. Citado na página 17.

Microsoft. *Introdução - Microsoft Threat Modeling Tool - Azure*. 2023. Disponível em: <<https://learn.microsoft.com/pt-br/azure/security/develop/threat-modeling-tool-getting-started>>. Citado 5 vezes nas páginas 7, 17, 26, 46 e 47.

Microsoft. *Microsoft Security Development Lifecycle Practices*. 2023. Disponível em: <<https://www.microsoft.com/en-us/securityengineering/sdl/practices>>. Citado 6 vezes nas páginas 16, 17, 22, 23, 24 e 25.

MONTE, D. P. R. d. Arquitetura de microsserviços: quando vale a pena migrar? fev. 2020. Accepted: 2020-07-28T22:20:15Z Publisher: Jaboaão dos Guararapes. Disponível em: <<https://repositorio.ifpe.edu.br/xmlui/handle/123456789/174>>. Citado na página 91.

MOORE, S. *Auto Scaling vs. Load Balancing: When to Use Each*. Disponível em: <<https://zesty.co/finops-academy/cloud-management/auto-scaling-vs-load-balancing/>>. Citado na página 91.

MUSCH, M. et al. ScriptProtect: Mitigating Unsafe Third-Party JavaScript Practices. In: *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*. Auckland New Zealand: ACM, 2019. p. 391–402. ISBN 978-1-4503-6752-3. Disponível em: <<https://dl.acm.org/doi/10.1145/3321705.3329841>>. Citado na página 82.

OWASP. *Input Validation - OWASP Cheat Sheet Series*. Disponível em: <[https://cheatsheetseries.owasp.org/cheatsheets/Input\\_Validation\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html)>. Citado na página 90.

OWASP. *Reverse Tabnabbing | OWASP Foundation*. Disponível em: <[https://owasp.org/www-community/attacks/Reverse\\_Tabnabbing](https://owasp.org/www-community/attacks/Reverse_Tabnabbing)>. Citado 2 vezes nas páginas 70 e 74.

- OWASP. *Threat Modeling* \textbar OWASP Foundation. 2010. Disponível em: <[https://owasp.org/www-community/Threat\\_Modeling](https://owasp.org/www-community/Threat_Modeling)>. Citado na página 25.
- OWASP. *ZAP – Getting Started*. 2023. Disponível em: <<https://www.zaproxy.org/getting-started/>>. Citado 5 vezes nas páginas 7, 17, 30, 31 e 32.
- PACHECO, D. R. P. Escalabilidade horizontal automática de serviços utilizando o componente HPA do Kubernetes. dez. 2023. Accepted: 2024-03-07T20:41:58Z Publisher: Serra. Disponível em: <<https://repositorio.ifes.edu.br/handle/123456789/4416>>. Citado na página 91.
- PIESSENS, F. The Cyber Security Body of Knowledge v1.1.0, 2021. In: . University of Bristol, 2021. Disponível em: <<https://www.cybok.org/>>. Citado na página 20.
- PIVETTA, L. Arquitetura de software: microsserviços. jan. 2023. Accepted: 2023-07-28T13:02:55Z Publisher: Universidade Federal de Santa Maria. Disponível em: <<http://repositorio.ufsm.br/handle/1/29798>>. Citado na página 92.
- RANSOME, J. F. et al. *Core software security: security at the source*. Boca Raton London New York: CRC Press, an Auerbach book, 2014. ISBN 978-1-4665-6095-6. Citado 2 vezes nas páginas 28 e 48.
- RIBEIRO, A. C. Análise estática de segurança de código-fonte : abordagem para avaliação de ferramentas. 2015. Disponível em: <<https://bdm.unb.br/handle/10483/11325>>. Citado 3 vezes nas páginas 29, 30 e 31.
- SANTOS, B. C. D. et al. Vulnerabilidade de Dados e a Percepção de Privacidade dos Usuários de Redes Sociais / Data Vulnerability and The Perception Privacy of Social Networks Users. *Brazilian Journal of Business*, v. 1, n. 4, p. 1728–1742, nov. 2019. ISSN 2596-1934. Disponível em: <<https://ojs.brazilianjournals.com.br/ojs/index.php/BJB/article/view/4836>>. Citado na página 20.
- SANTOS, M. G. D. UNIVERSIDADE FEDERAL DA PARAÍBA CENTRO DE INFORMÁTICA CURSO DE GRADUAÇÃO EM CIÊNCIA DE COMPUTAÇÃO. 2023. Citado na página 81.
- SAXENA, T. *Regular Expression Denial of Service (ReDoS) Attack: A Practical Guide*. 2024. Disponível em: <<https://blog.defensiumlabs.com/regular-expression-denial-of-service-redos-attack-a-practical-guide/>>. Citado na página 73.
- SCHWAB, K. *A Quarta Revolução Industrial*. [S.l.]: EDIPRO, 2019. ISBN 978-85-521-0046-1. Citado na página 15.
- SEBRAE. *O Impacto da pandemia de Coronavírus nos Pequenos Negócios – 13ª Edição do Sebrae – Dezembro* \textbar 2021. 2021. Publication Title: FGV Projetos. Disponível em: <<https://fgvprojetos.fgv.br/artigos/o-impacto-da-pandemia-de-coronavirus-nos-pequenos-negocios-13a-edicao-do-sebrae-dezembro>>. Citado na página 15.
- SHI, Z. et al. Threat Modeling Tools: A Taxonomy. *IEEE Security & Privacy*, v. 20, n. 4, p. 29–39, jul. 2022. ISSN 1558-4046. Citado 2 vezes nas páginas 26 e 27.

- SHOSTACK, A. *Threat Modeling: Designing for Security*. [S.l.]: John Wiley & Sons, 2014. ISBN 978-1-118-81005-7. Citado na página 25.
- SONAR. *Code Quality Tool & Secure Analysis with SonarQube*. 2023. Disponível em: <<https://www.sonarsource.com/products/sonarqube/>>. Citado na página 17.
- SONI, M. M. A Scientific Review of DAST (Dynamic Application Security Testing) and Its Specification. *International Journal of Engineering Research*, v. 1, n. 1, p. 01–04, jun. 2021. ISSN 2799-1873. Disponível em: <<http://www.ijeres.org/index.php/journal/article/view/1>>. Citado na página 30.
- SOUZA, M. *Comparativo de desempenho de um sistema de cache para aplicações web utilizando bancos de dados chave-valor*. Tese (Doutorado) — INSTITUTO FEDERAL DE SANTA CATARINA, São José - SC, 2022. Disponível em: <[https://wiki.sj.ifsc.edu.br/images/8/8a/TCC\\_55\\_MARINA\\_SOUZA.pdf](https://wiki.sj.ifsc.edu.br/images/8/8a/TCC_55_MARINA_SOUZA.pdf)>. Citado na página 65.
- SOUZA, R. C. R. d. *Adotando modelagem de ameaças em projetos ágeis de desenvolvimento de software*. 2023. Type: bachelorThesis. Disponível em: <<https://repositorio.ufpe.br/handle/123456789/49925>>. Citado na página 26.
- STAIKU, C.-A.; PRADEL, M.; DARMSTADT, T. Freezing the Web: A Study of ReDoS Vulnerabilities in JavaScript-based Web Servers. 2018. Citado na página 73.
- TENABLE. *Bootstrap 4.x < 4.3.1 Cross-Site Scripting*. 2019. Disponível em: <<https://www.tenable.com/plugins/was/112376>>. Citado na página 81.
- TRENDMICRO. *An in-depth HTTP Strict Transport Security Tutorial*. 2023. Section: expert perspective. Disponível em: <[https://www.trendmicro.com/en\\_us/devops/23/a/http-strict-transport-security-tutorial.html](https://www.trendmicro.com/en_us/devops/23/a/http-strict-transport-security-tutorial.html)>. Citado 2 vezes nas páginas 83 e 90.
- VAKALI, A.; PALLIS, G. Content delivery networks: status and trends. *IEEE Internet Computing*, v. 7, n. 6, p. 68–74, nov. 2003. ISSN 1941-0131. Conference Name: IEEE Internet Computing. Disponível em: <<https://ieeexplore.ieee.org/abstract/document/1250586>>. Citado na página 92.
- VIEIRA, M. D. F.; SILVA, C. M. S. D. A Educação no contexto da pandemia de COVID-19: uma revisão sistemática de literatura. *Revista Brasileira de Informática na Educação*, v. 28, p. 1013–1031, fev. 2020. ISSN 2317-6121, 1414-5685. Disponível em: <<https://br-ie.org/pub/index.php/rbie/article/view/v28p1013>>. Citado na página 15.
- WOHLIN, C. et al. Case Studies. In: WOHLIN, C. et al. (Ed.). *Experimentation in Software Engineering*. Berlin, Heidelberg: Springer, 2012. p. 55–72. ISBN 978-3-642-29044-2. Disponível em: <[https://doi.org/10.1007/978-3-642-29044-2\\_5](https://doi.org/10.1007/978-3-642-29044-2_5)>. Citado na página 33.
- XIN, T.; XIAOFANG, B. Online Banking Security Analysis based on STRIDE Threat Model. *International Journal of Security and Its Applications*, v. 8, n. 2, p. 271–282, mar. 2014. ISSN 17389976, 17389976. Disponível em: <[http://article.nadiapub.com/IJSIA/vol8\\_no2/28.pdf](http://article.nadiapub.com/IJSIA/vol8_no2/28.pdf)>. Citado na página 27.

# Apêndices

# APÊNDICE A – Primeiro Apêndice

Tabela 7 – Dados coletados na modelagem de ameaças

ID	Título	Categoria	Descrição	Interação	Prioridade
1	<i>Script</i> entre sites	Adulteração	O servidor web ' <i>Proxy Apache</i> ' pode estar sujeito a um ataque de <i>script</i> entre sites por não impedir entradas não confiáveis	HTTPS	Baixa
2	Elevação usando re-presentação	Elevação de privilégio	<i>Proxy apache</i> pode ser capaz de personificar o contexto do usuário humano a fim de ganhar mais privilégio	HTTPS	Baixa
14	Memória de processo <i>Proxy Apache</i> adulterada	Adulteração	Se o <i>Proxy Apache</i> tiver acesso à memória, como memória compartilhada ou ponteiros, ou tiver a capacidade de controlar o que o <i>Front</i> executa (por exemplo, passando para trás um ponteiro de função.), o apache do <i>proxy</i> poderá adulterar o <i>front</i> . Considere se a função poderia funcionar com menos acesso à memória, como passar dados em vez de ponteiros, copiar os dados fornecidos e, em seguida, validá-los	HTTP	Média
15	<i>Script</i> entre sites	Adulteração	O servidor web ' <i>Front</i> ' pode estar sujeito a um ataque de <i>script</i> entre sites por não impedir entradas não confiáveis	HTTP	Baixa
16	Elevação usando re-presentação	Elevação de privilégio	<i>Front</i> apache pode ser capaz de personificar o contexto do <i>Proxy Apache</i> a fim de ganhar mais privilégio	HTTP	Baixa

Continua na próxima página

ID	Título	Categoria	Descrição	Interação	Prioridade
17	<i>Script</i> entre sites	Adulteração	O servidor web ' <i>Proxy Apache</i> ' pode estar sujeito a um ataque de <i>script</i> entre sites por não impedir entradas não confiáveis	HTTP	Média
18	Elevação usando re-presentação	Elevação de privilégio	<i>Proxy Apache</i> pode ser capaz de personificar o contexto do <i>Front</i> a fim de ganhar mais privilégio	HTTP	Baixa
19	Elevação usando re-presentação	Elevação de privilégio	<i>Middleware</i> pode ser capaz de personificar o contexto do <i>Front</i> a fim de ganhar mais privilégio	HTTP	Baixa
20	Memória de processo <i>Middleware</i> adulterada	Adulteração	Se o <i>middleware</i> tiver acesso à memória, como memória compartilhada ou ponteiros, ou tiver a capacidade de controlar o que o <i>Front</i> executa (por exemplo, passando para trás um ponteiro de função.), o <i>middleware</i> poderá adulterar o <i>front</i> . Considere se a função poderia funcionar com menos acesso à memória, como passar dados em vez de ponteiros, copiar os dados fornecidos e, em seguida, validá-los	HTTP	Média
21	<i>Script</i> entre sites	Adulteração	O servidor web ' <i>Front</i> ' pode estar sujeito a um ataque de <i>script</i> entre sites por não impedir entradas não confiáveis	HTTP	Baixa
22	Elevação usando re-presentação	Elevação de privilégio	<i>Front</i> pode ser capaz de personificar o contexto do <i>Middleware</i> a fim de ganhar mais privilégio	HTTP	Média

Continua na próxima página

ID	Título	Categoria	Descrição	Interação	Prioridade
25	Potencial repúdio de dados no <i>Proxy Apache</i>	Repúdio	O <i>Proxy Apache</i> afirma que não recebeu dados de uma fonte fora do limite de confiança. Considere o uso de registro em <i>log</i> ou auditoria para registrar a fonte, o tempo e o resumo dos dados recebidos	HTTPS	Média
26	Falha ou parada de processo potencial para <i>proxy Apache</i>	Negação de serviço	<i>Proxy Apache</i> falha, para ou é executado lentamente; em todos os casos, viola uma métrica de disponibilidade	HTTPS	Média
27	Fluxo de dados HPPTS potencialmente interrompido	Negação de serviço	Um agente externo interrompe o fluxo de informações através do limite de confiança em qualquer direção	HTTPS	Média
28	<i>Proxy Apache</i> pode estar sujeito a elevação de privilégio usando a execução remota de código	Elevação de privilégio	Usuário humano pode ser capaz de executar remotamente código para <i>proxy Apache</i>	HTTPS	Baixa
29	Elevação alterando o fluxo de execução no <i>Proxy Apache</i>	Elevação de privilégio	Um invasor pode passar dados para o <i>Proxy Apache</i> para alterar o fluxo de execução do programa dentro do <i>Proxy Apache</i> para a escolha do invasor.	HTTPS	Média

Continua na próxima página



ID	Título	Categoria	Descrição	Interação	Prioridade
30	Falsificação de solicitação entre sites	Falsificação	A falsificação de solicitação entre sites (CSRF ou XSRF) é um tipo de ataque no qual um invasor força o navegador de um usuário a fazer uma solicitação forjada para um site vulnerável explorando uma relação de confiança existente	HTTPS	Média
31	Falsificação da entidade de destino externo do usuário humano	Falsificação	O Usuário Humano pode ser falsificado por um invasor e isso pode levar ao envio de dados para o alvo do invasor em vez do Usuário Humano. Considere o uso de um mecanismo de autenticação padrão para identificar a entidade externa	HTTPS	Média
32	Usuário humano de entidade externa potencialmente nega o recebimento de dados	Repúdio	O usuário humano alega que não recebeu dados de um processo do outro lado do limite de confiança. Considere o uso de <i>log</i> ou auditoria para registrar a origem, a hora e o resumo dos dados recebidos.	HTTPS	Média
33	Fluxo de dados HPPTS potencialmente interrompido	Negação de serviço	Um agente externo interrompe o fluxo de informações através do limite de confiança em qualquer direção	HTTPS	Média

Continua na próxima página

ID	Título	Categoria	Descrição	Interação	Prioridade
34	Falsificando a entidade externa do usuário humano	Falsificando	Usuário humano pode ser falsificado por um invasor e isso pode levar ao acesso não autorizado ao <i>Proxy Apache</i> . Considere o uso de um mecanismo de autenticação padrão para identificar a entidade externa	HPPTS	Alta
35	Elevação usando representação	Elevação usando	A API pode representar o contexto da APISIX para obter privilégios adicionais.	HTTP	Baixa
36	Elevação usando representação	Elevação usando	APISIX pode ser capaz de representar o contexto da API para obter privilégios adicionais.	HTTP	Média
37	Elevação usando representação	Elevação usando	O <i>middleware</i> pode ser capaz de representar o contexto do APISIX para obter privilégios adicionais.	HTTP	Média
38	Elevação usando representação	Elevação usando	A API pode representar o contexto da APISIX para obter privilégios adicionais.	HTTP	Baixa
39	Falsificação da entidade de destino externo do CFM da API	Falsificação	O CFM da API pode ser falsificado por um invasor e isso pode levar ao envio de dados para o destino do invasor em vez do CFM da API. Considere o uso de um mecanismo de autenticação padrão para identificar a entidade externa.	HPPTS	Média
40	API de entidade externa CFM potencialmente nega o recebimento de dados	Repúdio	A API CFM afirma que não recebeu dados de um processo do outro lado do limite de confiança. Considere usar <i>log</i> ou auditoria para registrar a origem, a hora e o resumo dos dados recebidos.	HPPTS	Média

Continua na próxima página

ID	Título	Categoria	Descrição	Interação	Prioridade
41	HTTPS de fluxo de dados é potencialmente interrompido	Negação de serviço	Um agente externo interrompe o fluxo de dados através de um limite de confiança em qualquer direção.	HPPTS	Média
42	Falsificando a entidade externa CFM da API	Falsificação	O CFM da API pode ser falsificado por um invasor e isso pode levar ao acesso não autorizado à API. Considere o uso de um mecanismo de autenticação padrão para identificar a entidade externa.	HPPTS	Média
43	Potencial repúdio de dados pela API	Repúdio	A API afirma que não recebeu dados de uma fonte fora do limite de confiança. Considere o uso de <i>log</i> ou auditoria para registrar a origem, a hora e o resumo dos dados recebidos.	HPPTS	Média
44	Falha ou parada potencial do processo para API	Negação de serviço	A API trava, interrompe, para ou é executada lentamente; em todos os casos, violando uma métrica de disponibilidade.	HPPTS	Alta
45	HTTPS de fluxo de dados é potencialmente interrompido	Negação de serviço	Um agente externo interrompe o fluxo de dados através de um limite de confiança em qualquer direção.	HPPTS	Alta
46	Elevação usando representação	Elevação de privilégio	A API pode representar o contexto do CFM da API para obter privilégios adicionais.	HPPTS	Média

Continua na próxima página

ID	Título	Categoria	Descrição	Interação	Prioridade
47	API pode estar sujeita a elevação de privilégio usando execução remota de código	Elevação de privilégio	O API CFM pode ser capaz de executar remotamente código para API.	HPPTS	Média
48	Elevação alterando o fluxo de execução na API	Elevação de privilégio	Um invasor pode passar dados para a API para alterar o fluxo de execução do programa dentro da API à escolha do invasor.	HPPTS	Média
49	Falsificação da entidade de destino externo CRF da API	Falsificação	O CRF DA API pode ser falsificado por um invasor e isso pode levar ao envio de dados para o destino do invasor em vez do CRF da API. Considere o uso de um mecanismo de autenticação padrão para identificar a entidade externa.	HPPTS	Média
50	CRF da API de entidade externa potencialmente nega o recebimento de dados	Repúdio	A API CRF alega que não recebeu dados de um processo do outro lado do limite de confiança. Considere o uso de <i>log</i> ou auditoria para registrar a origem, a hora e o resumo dos dados recebidos.	HPPTS	Média
51	HTTPS de fluxo de dados é potencialmente interrompido	Negação de serviço	Um agente externo interrompe o fluxo de dados através de um limite de confiança em qualquer direção.	HPPTS	Média

Continua na próxima página

ID	Título	Categoria	Descrição	Interação	Prioridade
52	Falsificação da entidade externa CRF da API	Falsificação	A CRF da API pode ser falsificada por um invasor e isso pode levar ao acesso não autorizado à API. Considere o uso de um mecanismo de autenticação padrão para identificar a entidade externa.	HPPTS	Média
53	Potencial repúdio de dados pela API	Repúdio	A API afirma que não recebeu dados de uma fonte fora do limite de confiança. Considere o uso de <i>log</i> ou auditoria para registrar a origem, a hora e o resumo dos dados recebidos.	HPPTS	Média
54	Falha ou parada potencial do processo para API	Negação de serviço	A API trava, interrompe, para ou é executada lentamente; em todos os casos, violando uma métrica de disponibilidade.	HPPTS	Alta
55	HTTPS de fluxo de dados é potencialmente interrompido	Negação de serviço	Um agente externo interrompe o fluxo de dados através de um limite de confiança em qualquer direção.	HPPTS	Média
56	Elevação usando representação	Elevação de privilégio	A API pode representar o contexto da CRF da API para obter privilégios adicionais.	HPPTS	Média
57	API pode estar sujeita a elevação de privilégio usando execução remota de código	Elevação de privilégio	O CRF DA API pode ser capaz de executar remotamente o código da API.	HPPTS	Média

Continua na próxima página

ID	Título	Categoria	Descrição	Interação	Prioridade
58	Elevação alterando o fluxo de execução na API	Elevação de privilégio	Um invasor pode passar dados para a API para alterar o fluxo de execução do programa dentro da API à escolha do invasor.	HTTPS	Média
59	Falsificação da entidade de destino externo CRO da API	Falsificação	A API CRO pode ser falsificada por um invasor e isso pode fazer com que os dados sejam enviados ao alvo do invasor em vez da API CRO. Considere usar um mecanismo de autenticação padrão para identificar a entidade externa.	HTTPS	Média
60	CRO de API de entidade externa potencialmente nega o recebimento de dados	Repúdio	A API CRO afirma que não recebeu dados de um processo do outro lado do limite de confiança. Considere o uso de <i>log</i> ou auditoria para registrar a origem, a hora e o resumo dos dados recebidos.	HTTPS	Média
61	HTTPS de fluxo de dados é potencialmente interrompido	Negação de serviço	Um agente externo interrompe o fluxo de dados através de um limite de confiança em qualquer direção.	HTTPS	Média
62	Falsificação da entidade externa CRO da API	Falsificação	A CRO da API pode ser falsificada por um invasor e isso pode levar ao acesso não autorizado à API. Considere o uso de um mecanismo de autenticação padrão para identificar a entidade externa.	HTTPS	Média

Continua na próxima página

ID	Título	Categoria	Descrição	Interação	Prioridade
63	Potencial repúdio de dados pela API	Repúdio	A API afirma que não recebeu dados de uma fonte fora do limite de confiança. Considere o uso de log ou auditoria para registrar a origem, a hora e o resumo dos dados recebidos.	HTTPS	Média
64	Falha ou parada potencial do processo para API	Negação de serviço	A API trava, interrompe, para ou é executada lentamente; em todos os casos, violando uma métrica de disponibilidade.	HTTPS	Alta
65	HTTPS de fluxo de dados é potencialmente interrompido	Negação de serviço	Um agente externo interrompe o fluxo de dados através de um limite de confiança em qualquer direção.	HTTPS	Alta
66	Elevação usando representação	Elevação de privilégio	A API pode representar o contexto da CRO da API para obter privilégios adicionais.	HTTPS	Média
67	API pode estar sujeita a elevação de privilégio usando execução remota de código	Elevação de privilégio	API CRO pode ser capaz de executar remotamente código para API.	HTTPS	Média
68	Elevação alterando o fluxo de execução na API	Elevação de privilégio	Um invasor pode passar dados para a API para alterar o fluxo de execução do programa dentro da API à escolha do invasor.	HTTPS	Média

## ANEXO A – Primeiro Anexo

Termo de Permissão para Divulgação de Informações sobre Vulnerabilidades do Serviço VALIDAR.

O Termo de Permissão para Divulgação de Informações sobre Vulnerabilidades do Serviço VALIDAR foi assinado pelo responsável pelo VALIDAR na época deste estudo de caso, Thiago Figaro Krasauskas, e pela autora deste trabalho, Carla Rocha Cangussú, discente da Universidade de Brasília. A assinatura foi realizada eletronicamente por meio do assinador digital fornecido pelo ITI<sup>1</sup>.

Devido a limitações da ferramenta de edição de texto utilizada neste estudo, as representações visuais das assinaturas não foram renderizadas.

---

<sup>1</sup> Disponível em: <https://assinador.iti.br/assinatura/index.xhtml>



# **Termo de Permissão para Divulgação de Informações sobre Vulnerabilidades do Serviço VALIDAR**

## **1. Identificação das Partes:**

### **Titular do Serviço:**

Eu, Thiago Figaro Krasauskas, responsável pelo VALIDAR - Serviço de Assinatura Eletrônica, prestado pelo Instituto Nacional da Informação (ITI).

### **Divulgante:**

Carla Rocha Cangussú, discente da Universidade de Brasília.

## **2. Objeto:**

O presente Termo tem por objeto a autorização do Titular do Serviço para que Carla Rocha Cangussú possa divulgar o nome do serviço VALIDAR e as informações detalhadas sobre as vulnerabilidades encontradas na aplicação, assim como sobre as respectivas ações de mitigação, em seu trabalho de conclusão de curso.

## **3. Autorização:**

O Titular do Serviço autoriza expressamente Carla Rocha Cangussú a:

- Divulgar o nome do serviço VALIDAR e as informações sobre as vulnerabilidades encontradas na aplicação e das respectivas ações de mitigação.
- Publicar as informações detalhadas sobre as vulnerabilidades no contexto de seu trabalho de conclusão de curso.

## **4. Responsabilidade:**

O Titular do Serviço isenta a Divulgante de qualquer responsabilidade por danos diretos ou indiretos que possam surgir da divulgação das informações sobre as vulnerabilidades, desde que a divulgação esteja em conformidade com os termos deste acordo.

## **5. Disposições Gerais:**

- Qualquer modificação deste Termo somente será válida se feita por escrito e assinada por ambas as partes.

- Este Termo é regido pelas leis da República Federativa do Brasil.
- Fica eleito o foro da Comarca de [Cidade/UF] para dirimir quaisquer controvérsias decorrentes deste Termo.

6. Assinaturas:

---

Thiago Figaro Krasauskas  
Servidor  
Instituto Nacional da Informação (ITI)

---

Carla Rocha Cangussú  
Discente  
Universidade de Brasília

Data:04/06/2023

---

## ANEXO B – Segundo Anexo

Comprovação da validade das assinaturas do Termo de Permissão para Divulgação de Informações sobre Vulnerabilidades do Serviço VALIDAR. A validação das assinatura foi realizada no VALIDAR e como demonstrado no anexo, as duas assinatura são válidas.



### Informações gerais do arquivo:

**Nome do arquivo:** TermoCarlaTCC\_assinado.pdf  
**Hash:** 0008180b3171afc5a338141130b27d5900d8f1d9aa21b4cb4e8f77478006cb74  
**Data da validação:** 11/09/2024 13:40:02 BRT

#### ✓ Informações da Assinatura:

**Assinado por:** THIAGO FIGARO KRASAUSKAS  
**CPF:** \*\*\*.891.818-\*\*  
**Nº de série de certificado emitente:** 0x1f75d6ae6056de33026c7994  
**Data da assinatura:** 05/06/2024 13:29:54 BRT



Assinatura aprovada.

#### ✓ Informações da Assinatura:

**Assinado por:** CARLA ROCHA CANGUSSU  
**CPF:** \*\*\*.388.685-\*\*  
**Nº de série de certificado emitente:** 0x3363d05c89441465  
**Data da assinatura:** 10/09/2024 08:48:49 BRT



Assinatura aprovada.

[Ver Relatório de Conformidade](#)

### ACESSO RÁPIDO

[Validar](#)

[Sobre](#)

[Dúvidas](#)

[Informações](#)

[Fale Conosco](#)