Superfície de ataque em SaaS: Impacto do OSINT na autenticação e mitigações

Pedro Augusto Paião[1], **Carlos Alberto da Silva**[2]

[1]Curso de Bacharelado em Sistemas de Informação (FACOM)
[2]Faculdade de Computação (FACOM)

Abstract. *This study examines how open-source intelligence (OSINT)[21] affects the login surface of a Software-as-a-Service (SaaS)[25] platform. Using OSINT, the study obtained a valid account identifier and executed controlled online password guessing[27] against an authorized account. The analysis covered login error-message consistency (user-not-found vs. wrong-password), the presence of basic rate limiting or lockout[27, 25], and the practical effort (attempts and time) until a clear defensive reaction. Results indicate that public data reduces the search space and can increase success odds when defenses are weak. The study presents actionable mitigations for authentication hardening.*

Resumo. *Este estudo avalia o impacto do open-source intelligence (OSINT)[21] na superfície de login de um ambiente Software-as-a-Service (SaaS)[25]. Mediante OSINT, o estudo obteve um identificador válido e realizou tentativas controladas de senha em conta autorizada. A análise contemplou a consistência das mensagens de erro (usuário inexistente vs. senha incorreta), a presença de rate limiting/bloqueio básico[27] e o esforço prático (tentativas e tempo) até reação defensiva. Os resultados indicam que dados públicos reduzem o espaço de busca e elevam a probabilidade de sucesso quando as defesas são fracas. O estudo apresenta mitigações práticas para o endurecimento da autenticação.*

1. Introdução

A digitalização acelerada de processos de negócio e a popularização do modelo *Software-as-a-Service* (SaaS) [25] ampliaram o perímetro de exposição de dados das organizações na Internet. Em paralelo, o volume de informações públicas e livremente acessíveis — *open-source intelligence* (OSINT) [21] — cresceu de forma significativa, como objetos armazenados do tipo: páginas institucionais, buscadores, metadados, repositórios, logs de certificados, índices de motores de busca, e outros. Esse contexto cria um elo direto entre dados publicamente expostos e a superfície de autenticação de aplicações web: quando um atacante, a partir de OSINT, reduz o espaço de busca de contas e credenciais, os controles de defesa no ponto de *login* tornam-se ainda mais críticos.

Embora a literatura enfatize o endurecimento de senhas, autenticação multifator (MFA) e monitoramento de abuso [4, 13], persistem fragilidades operacionais: mensagens de erros que permitem inferir a existência de usuários, ausência de *rate limiting* e *lockouts* [27, 25] efetivos, ou políticas de recuperação de conta que contornam barreiras do *login* principal. Na prática, o uso de OSINT pode transformar um ataque de *online password guessing* [27] em um processo mais curto, barato e silencioso, especialmente em ambientes SaaS com grande base de usuários e exposição pública de padrões de *e-mail* corporativo.

O presente artigo apresenta um *pentest* do tipo superfície de ataque em SaaS por meio do impacto do OSINT no processo de autenticação e mitigações.

2. Trabalhos relacionados

O *framework* proposto por Macêdo para inteligência cibernética [18], baseada em fontes abertas (OSINT) oferece um encadeamento disciplinado de atividades (coleta, triagem, análise e disseminação) que dialoga diretamente com a metodologia adotada neste estudo. Na etapa de **coleta**, o uso de buscadores, CT logs e serviços de agregação, por exemplo, busca de padrões institucionais de *e-mail*, apoia a redução do espaço de busca na autenticação de ambientes *SaaS*; na **triagem**, a higienização/-validação passiva mitiga vieses e falsos positivos; na **análise**, a operacionalização em métricas — consistência de mensagens de erro, presença/limiar de *rate limiting*, esforço temporal até reação e o *enforcement* de MFA — permite comparar controles e priorizar mitigações; por fim, a **disseminação** estrutura a documentação reprodutível dos achados, como *prints*, tabelas e fluxos, e o registro ético de evidências. Consequentemente, o arcabouço de Macêdo fornece base conceitual e processual para transformar OSINT em insumos mensuráveis e acionáveis no endurecimento do processo de *login*.

Correlação com este TCC. Enquanto Macêdo oferece um arcabouço processual para transformar OSINT em inteligência operacional [18], este TCC aplica e estende esse arcabouço ao domínio específico da autenticação em *SaaS*, aportando evidências empíricas e métricas reproduzíveis: (i) demonstra, na fase de coleta/triagem, que fontes abertas reduzem substancialmente o espaço de busca de identificadores (e-mails) com menor custo temporal que a enumeração por endpoint; (ii) na análise, quantifica a "granularidade" de erros (diferenças 200/401 no *recovery*, ausência de `429` no *login* amostrado) e estima esforço temporal sob concorrência por *threads*; e (iii) na disseminação, apresenta artefatos auditáveis (fluxograma do procedimento, capturas do Fiddler, tabelas e listagens de terminal) que fecham o ciclo "achado $\rightarrow$ métrica $\rightarrow$ mitigação". Assim, contribui ao estado da arte ao ancorar o framework em um caso aplicado com indicadores objetivos e um mapa de mitigações alinhado a boas práticas de autenticação segura.

3. Fundamentação Teórica

Este capítulo define o escopo do estudo e apresenta os conceitos que sustentam a análise da superfície de autenticação em ambiente *Software-as-a-Service* (SaaS) combinada ao uso de *open-source intelligence* (OSINT).

3.1. OSINT: definição e escopo

Open-Source Intelligence (OSINT) refere-se à coleta e análise de informações publicamente acessíveis, obtidas sem autenticação privilegiada e sem contornar controles técnicos [21]. Exemplos incluem: páginas institucionais, mecanismos de busca, metadados, repositórios públicos, logs de transparência de certificados e índices de buscadores. No contexto deste estudo, OSINT é empregado para reduzir o espaço de busca na fase de reconhecimento, com foco em identificar padrões de endereçamento e candidatos a identificadores de conta, como por exemplo os *e-mails* corporativos.

3.2. Superfície de autenticação em SaaS

Aplicações SaaS expõem pontos de entrada de autenticação a uma ampla base de usuários e a tráfego da Internet. A literatura e normas de referência recomendam controles, como senhas robustas, autenticação multifator (MFA), tratamento consistente de erros e mecanismos antiabuso [13, 25, 26]. Em particular, a camada de autenticação deve reduzir canais que facilitem enumeração de contas, impor políticas de senha compatíveis com verificação contra listas comprometidas e aplicar limites de taxa e bloqueios temporários sob tentativas repetidas.

3.3. Enumeração de contas e mensagens de erro

Enumeração de contas ocorre quando o sistema vaza, direta ou indiretamente, a existência de um usuário, por exemplo, mensagens distintas "usuário inexistente" vs. "senha incorreta", ou variações de código/latência.

A **consistência** das mensagens de erro — às vezes chamada de "granularidade das mensagens" — é central: quanto mais específicas e divergentes, maior o potencial de enumeração. A recomendação é padronizar as respostas de falhas de autenticação de forma indistinguível ao atacante [25].

3.4. Ataques de senha: definições úteis

No contexto de tentativas *online* por meio da própria aplicação alvo, diferenciam-se [26, 27] na formas de *pentest*:

- ***Online password guessing/brute force***: múltiplas senhas tentadas contra um único usuário;
- ***Password spraying***: uma (ou poucas) senha(s) tentada(s) contra muitos usuários;
- ***Credential stuffing***: reaproveitamento de pares usuário/senha obtidos em vazamentos anteriores, testando em outro(s) serviço(s).

Cada modalidade de *pentest*, demanda telemetria e contramedidas específicas que podem variar para limites por conta, por IP/ASN, por dispositivo, e por padrão comportamental.

3.5. Controles recomendados

As Normas e guias de boas práticas propõem um conjunto de controles que, combinados, reduzem o sucesso de ataques de senha [13, 25]:

- **Mensagens de erro genéricas e consistentes** para qualquer falha de *login*, evitando enumeração;
- ***Rate limiting*** com janela deslizante, atrasos progressivos e **bloqueio temporário** após N falhas, por conta e/ou por origem;
- **MFA** obrigatório por risco ou função sensível, com verificação de senha contra listas comprometidas;
- **Detecção de padrões de abuso**, como *spraying e stuffing*, alertas e registros com correlação para [usuário, IP, ASN, agente, dispositivo].

3.6. Métricas de interesse

Para avaliar a efetividade defensiva neste cenário, destacam-se os critérios de: (i) consistência das mensagens de erro; (ii) detecção de *rate limiting*/bloqueio e respectivos limiares; (iii) esforço prático (número de tentativas e tempo) até reação defensiva; e (iv) presença e enforcement de MFA. Tais métricas conectam os achados com as mitigações, e permitem acompanhar a evolução ao longo do tempo [13, 25].

4. Ferramentas

Para apoiar o reconhecimento assistido por OSINT, o mapeamento do fluxo de autenticação e as tentativas online controladas, utilizamos o seguinte conjunto de ferramentas, a seguir.

4.1. *Oracle VirtualBox*

Oracle VirtualBox [23] é um *hipervisor* de uso geral para criação e execução de máquinas virtuais. Foi empregado para isolar o ambiente de testes e garantir reprodutibilidade, com controle de rede e *snapshots* para retorno a estados conhecidos.

4.2. *Kali Linux*

Kali Linux [22] é uma distribuição baseada em *Debian* voltada a segurança, usada como sistema operacional da máquina virtual. Serviu como base para as ferramentas de reconhecimento e apoio a OSINT, por exemplo, os comandos: `amass`, `theHarvester`, e `subfinder` para os utilitários de linha de comando.

4.3. *Censys* (apoio OSINT para superfície exposta)

Plataforma de busca em ativos/serviços expostos na *Internet*, a partir de *scans* contínuos [6]. Foi utilizada para identificar *hosts*, certificados de autenticações e metadados públicos associados ao domínio-alvo. Ferramentas equivalentes ou complementares para o processo de mitigação, incluem: *Shodan, ZoomEye e BinaryEdge*. [34, 15, 3]

4.4. *Fiddler* (proxy HTTP para inspeção)

Ferramenta de *proxy* e depuração HTTP(S) [35] utilizada para observar as requisições do aplicativo (web ou móvel), mapear rotas de API, verbos, cabeçalhos, parâmetros e padrões de resposta/erro. Alternativas com função semelhante incluem *Burp Suite* [28] e *mitmproxy* [20]. Este mapeamento guiou os cenários de teste do processo de *login*.

4.5. *Python 3* + `requests` (script de tentativas controladas)

Foi desenvolvido um *script* em *Python 3* [31], usando a biblioteca `requests` [32], para executar tentativas *online* controladas de autenticação contra uma conta autorizada cujo identificador foi obtido via OSINT. O *script* de *pentest* registra por tentativa: tempo de resposta, código de status, trecho da mensagem de erro e contadores até a primeira reação defensiva clara, como exemplo para bloqueio/atraso. Os resultados são gravados em arquivos CSV para análise posterior.

4.6. *Crunch* (geração de *wordlists*)

O `crunch` [10] foi empregado para gerar *wordlists* determinísticas, a partir de alfabetos e comprimentos definidos, a fim de testar políticas e comportamentos básicos do ponto de autenticação.

Exemplo (somente dígitos, 8 caracteres):

```
crunch 8 8 0123456789 -o wordlist_8digitos.txt
```

Devido à explosão combinatória, evitou-se listas muito grandes, como exemplo, alfanumérico com 8+ posições. Quando necessário, utilizou-se `-c` (divisão em blocos) ou compressão em fluxo para gerenciamento de disco.

4.7. Utilitários de apoio

Além das ferramentas principais, o estudo adotou utilitários de linha de comando para *orquestrar* requisições pontuais, inspecionar respostas e higienizar artefatos (*logs*, CSVs). O uso desses utilitários teve caráter operacional, visando reprodutibilidade e rastreabilidade dos testes, sem substituir as medições centrais.

Funções e papéis.

`curl / httpie` Emissão de requisições HTTP(s) pontuais (ex.: `POST` com JSON), captura de código de status e cabeçalhos para validar rapidamente endpoints mapeados ([11, 14]).

`jq` Inspeção e extração de campos em respostas JSON, facilitando validação de mensagens, *payloads* e estruturas retornadas ([16]).

`grep/awk` Filtragem e sumarização de arquivos de *log*/CSV (ex.: contagem de códigos HTTP, amostragens e cortes por tempo) ([12, 33]).

Exemplos rápidos.

(i) Requisição de teste no *login* com `curl`:

```
curl -s -o /tmp/resp.json -w "%{http_code}\n" \
  -H "Content-Type: application/json" \
  -X POST "https://<host>/api/auth/login" \
  --data '{"email":"usuario@dominio","password":"SENHA-TESTE"}'
```

(ii) Equivalente com `httpie`, exibindo resposta formatada:

```
http POST https://<host>/api/auth/login \
  Content-Type:application/json \
  email=usuario@dominio password=SENHA-TESTE
```

(iii) Extração de campo com `jq` (ex.: mensagem de erro):

```
cat /tmp/resp.json | jq -r '.message // .error // "sem-mensagem"'
```

(iv) Contagem simples de códigos em CSV gerado pelo script:

```
# cabecalho: timestamp,attempt,http_status,elapsed_ms,snippet
cut -d, -f3 login_results.csv | tail -n +2 | sort | uniq -c
```

Observação. Esses utilitários foram empregados apenas como *apoio* para checagens rápidas e organização dos artefatos de experimento (por exemplo, confirmação de que a mensagem de falha permaneceu uniforme ou de que não houve retorno `429` na amostra). As conclusões do estudo derivam das seções metodológicas e dos resultados consolidados, não de testes ad hoc.

4.8. Ferramentas OSINT adicionais (quando aplicável)

Além do Censys, foram úteis as seguintes ferramentas:

- `theHarvester` [8]: para coleta de *e-mails* e artefatos públicos a partir de múltiplas fontes.
- `amass` [24]/`subfinder` [30]: para enumeração de subdomínios via fontes abertas, certificados de autenticações e resoluções DNS passivas.

4.9. Reprodutibilidade e limites

Todo o conjunto foi executado em ambiente isolado (VM), com limite de taxa no *script* em *Python*[31] para evitar saturação de serviço. Não foram usados *dumps* privados, credenciais vazadas ou técnicas que contornassem controles técnicos; as tentativas foram restritas a uma conta autorizada e em janela de tempo controlada.

5. Metodologia

Esta seção descreve, de forma reprodutível, o procedimento adotado para avaliar o impacto do *open-source intelligence* (OSINT) sobre a superfície de autenticação de uma aplicação *Software-as-a-Service* (SaaS). O experimento considera um cenário autorizado em que um identificador de conta, por exemplo, o *e-mail*, é obtido por fontes abertas, sobre o qual são realizadas tentativas *online* controladas de autenticação, com coleta de métricas operacionais do processo de *login*. O método prioriza baixo impacto no alvo, limitação de taxa e registro consistente dos resultados para posterior análise.

5.1. Passo 1 — Ativação do proxy HTTPS e mapeamento inicial

O primeiro passo consistiu na ativação de um proxy HTTPS (*Fiddler* [35]) para inspeção do tráfego da aplicação e identificação dos pontos de entrada relevantes. Com a interceptação TLS habilitada, o domínio de interesse foi confirmado e dois *endpoints* chave da API de autenticação foram mapeados: `/api/auth/login` e `/api/auth/recovery-password`.

Procedimento.

1. Configuração do*proxy* HTTPS no *host* de testes para inspeção TLS e instalação do certificado raiz.
2. Direcionamento do tráfego do aplicativo por meio do *proxy* para sistema/emulador/navegador.
3. Reprodução do fluxo de autenticação com registro de requisições e respostas.
4. Filtragem por *host* e métodos (`POST`/`GET`) para localizar os *endpoints*.
5. Salvamento de exemplos de *payloads*, cabeçalhos e códigos de *status* para padronização dos testes.

Artefatos coletados.

- **Endpoint de login** (`/api/auth/login`): método, cabeçalhos `Content-Type`, estrutura de corpo (exemplo: `email`, `password`) e padrão de resposta/erro.
- **Endpoint de recuperação** (`/api/auth/recovery-password`): método, parâmetros esperados (exemplo: `email`) e respostas.
- **Domínio/porta**: `***.com.br:****`, para validação e repetibilidade dos testes.

Conforme ilustrado nas Figuras 1 e 2, o `Fiddler` foi utilizado para inspecionar o tráfego legítimo entre cliente e API. A Figura 1 mostra a captura de uma requisição para o endpoint de recuperação de senha `/api/auth/recovery-password`, destacando o método HTTP, a URL completa e o corpo JSON contendo apenas o campo `"email"`

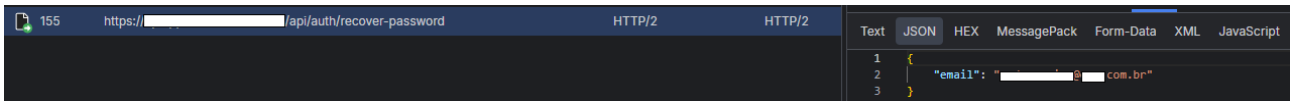

Figura 1. Rota de recuperar senha com o corpo enviado (`/api/auth/recovery-password`).

A Figura 2 apresenta a requisição para o endpoint de *login* `/api/auth/login`, na qual, além do `"email"`, são enviados os demais parâmetros de autenticação (por exemplo, senha e tipo de *grant*). Essas capturas serviram de base para reproduzir o mesmo padrão de chamadas nos testes controlados com outras ferramentas.

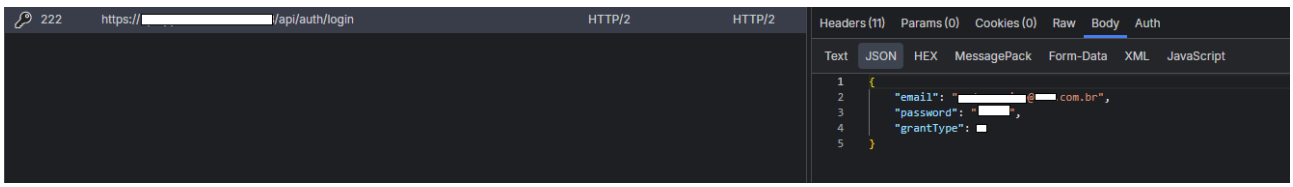

Figura 2. Rota de *login* com o corpo enviado (`/api/auth/login`).

5.2. Passo 2 — Reconhecimento de infraestrutura e indícios via OSINT

Após o mapeamento de *endpoints*, foi realizado um reconhecimento da camada de exposição para caracterizar o *front* do domínio e verificar a presença de WAF/CDN. A sequência adotada incluiu varredura com `nmap` [17] (portas/serviços e metadados TLS/SNI), consulta a `whois` (propriedade/ASN) e busca no *Censys* [6] (OSINT) para levantar possíveis IPs candidatos por correlação de certificado/*hostname*.

Procedimento.

1. **`nmap`**: confirmação de portas expostas e coleta de metadados.
2. **`whois`**: verificação de propriedade/ASN, com indicação de *Cloudflare* como WAF/CDN.
3. ***Censys***: correlação por certificados/*hostnames* para identificação de IPs candidatos.

Interpretação. A combinação de *nmap* [17] e *whois* indica roteamento pelo *edge* de CDN/WAF (Cloudflare), o que sugere proteções padrão (*rate limiting*, *bot management*) [26]. A consulta no *Censys* [6] fornece indícios de IP de origem por correlação de certificado, mantendo o escopo não intrusivo do estudo.

Na Figura 3 é apresentado o resultado de um *port scan* com a ferramenta `nmap` sobre o domínio em estudo. O comando revela que o host está ativo e mostra apenas quatro portas TCP abertas (80, 443, 8080 e 8443), associadas a serviços HTTP/HTTPS e *http-proxy*, enquanto as demais portas foram filtradas. Além das portas e serviços, o `nmap` exibe o endereço IP atualmente publicado para o domínio e outros endereços correlatos não varridos, além de metadados coletados na negociação TLS/SNI, que serão reutilizados nas etapas seguintes do reconhecimento.

Riscos e implicações de portas padrão (80/443). Mesmo sob CDN/WAF, a exposição de **80/tcp** e **443/tcp** impõe cuidados específicos no ponto de autenticação e no *edge*:

- **Porta 80 (HTTP claro)**: exige redirecionamento imediato e universal para HTTPS com **HSTS** (incluindo `includeSubDomains` e, idealmente, `preload`); ausência ou exceções criam risco de *downgrade*, *cookie* sem `Secure` e exposição de *tokens* em texto claro [25, 7].
- **Porta 443 (TLS)**: requer política de **TLS forte** (desativar TLS 1.0/1.1, *ciphers* fracos e `NULL`/`EXPORT`), `Secure`/`HttpOnly`/`SameSite` em *cookies* de sessão, `Strict-Transport-Security` e proteção contra **CSRF**/CORS permissivo [25].
- **Coerência entre portas/hosts**: caminhos alternativos (8080/8443, `www` vs. raiz, IP direto) devem aplicar **as mesmas políticas** de autenticação, mensagens de erro e *rate limiting*; inconsistências permitem *bypass* de controles e facilitam enumeração [26].
- **Cabeçalhos e verbos inseguros**: desabilitar `TRACE`/`TRACK`, validar *verbs* aceitos, aplicar `Content-Security-Policy`, `X-Content-Type-Options`, `X-Frame-Options`/`Frame-Options`, `Referrer-Policy` [25].
- **Ataques de camada HTTP**: atenção a *request smuggling*/*desync*, *cache poisoning*, diferenças HTTP/1.1 vs. HTTP/2 (e HTTP/3/QUIC), e regras específicas no WAF/CDN para *login* e *recovery* [26].
- **Superfície de descoberta**: rotas padrões em 80/443 (p. ex., `/.well-known/`, `/.git`, `/.env`, `/metrics`) devem estar bloqueadas no *edge*/origin; exposição indevida acelera OSINT e exploração [7].
- **Conformidade e privacidade**: endurecimento de transporte e telemetria de abuso no ponto de *login* contribuem para princípios de segurança por padrão e por projeto (LGPD/ANPD) [2].

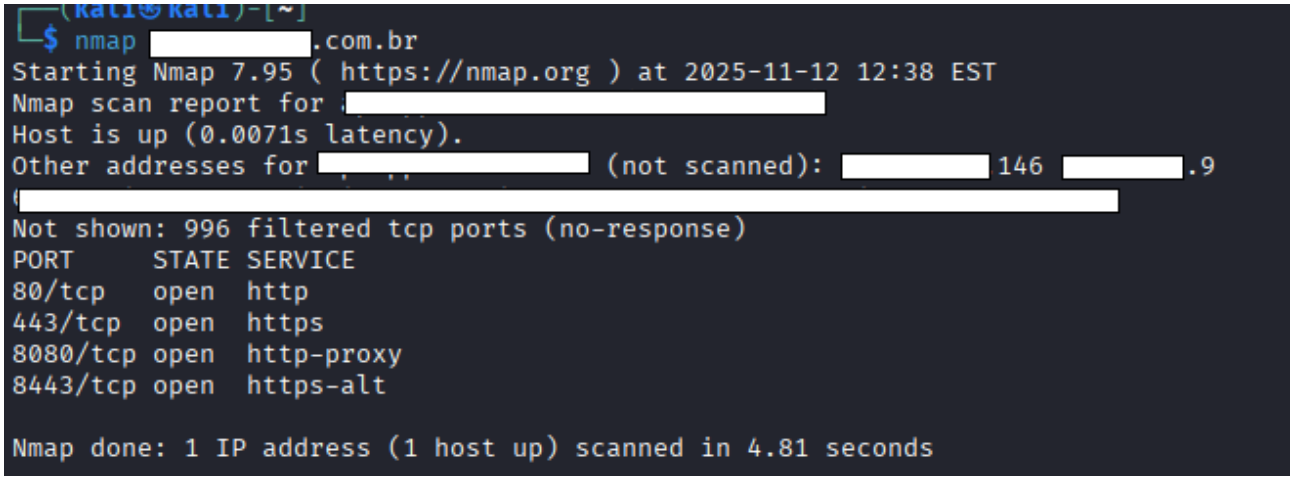

Figura 3. `nmap`: portas/serviços e metadados TLS/SNI observados para o domínio.

Na Figura 4 é mostrado o resultado de uma consulta `whois` para o mesmo endereço IP. Os campos `NetRange`, `CIDR` e `NetName` indicam que o bloco pertence à rede `CLOUDFLARENET`, administrada pela Cloudflare, Inc. Também são exibidas informações de organização, sistema autônomo e datas de registro/atualização do bloco. Esses dados confirmam que o endereço identificado no `nmap` está atrás de uma camada de CDN/WAF, ou seja, não corresponde necessariamente ao servidor de aplicação de origem.

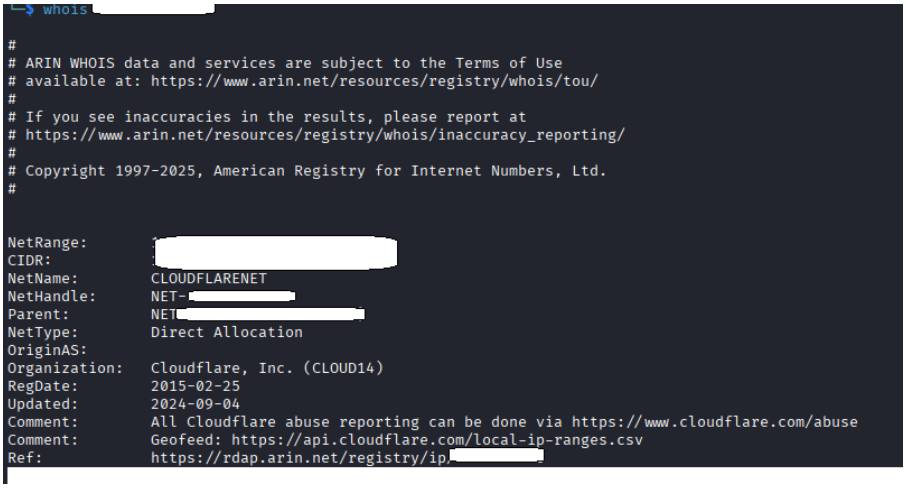

Figura 4. `whois`: evidência de CDN/WAF (*Cloudflare*) como camada de exposição.

A Figura 5 apresenta a interface da plataforma *Censys* filtrando hosts relacionados ao domínio analisado. A busca retorna uma lista de endereços IP candidatos, cada um com informações de certificado TLS/hostname, sistema operacional, provedor (AS), localização geográfica e portas/serviços expostos (por exemplo, 443/HTTPs em diferentes provedores brasileiros). A correlação entre o nome do certificado, o domínio e os blocos de IP fora da rede `CLOUDFLARENET` permite inferir possíveis IPs reais da aplicação, ilustrando o papel de fontes abertas (*OSINT*) na redução da superfície de incerteza durante o reconhecimento.

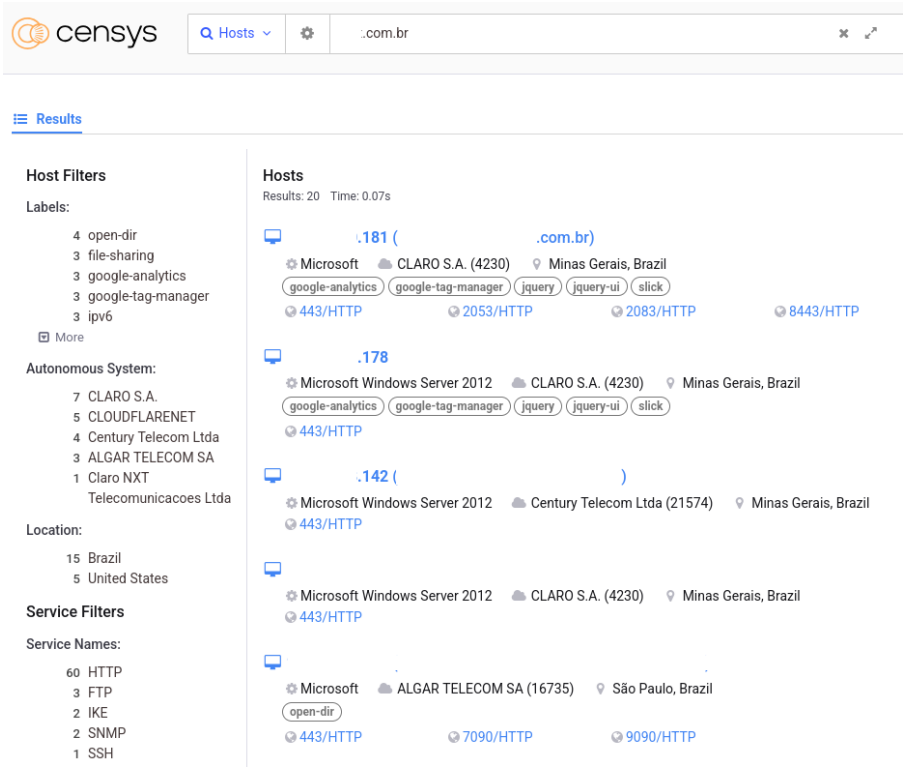

Figura 5. *Censys*: correlação por certificado/hostname indicando possíveis IPs candidatos (OSINT).

Cuidados e limitações.

- **Falsos positivos:** certificados compartilhados e SNI semelhantes podem levar a correlações equivocadas; tratar sempre como indício.
- **Mutabilidade:** mapeamentos em CDN/WAF são dinâmicos; resultados refletem o período observado.
- **Escopo:** não foram realizados testes para contornar o WAF nem varreduras no possível IP de origem; o estudo permanece no ponto de autenticação autorizado.

5.3. Passo 3 — Validação do *endpoint* no IP identificado (*Postman*)

Com o IP/porta identificados, a validação do ponto de autenticação foi executada por meio do envio, no *Postman* [29], da mesma requisição de *login* mapeada no *Fiddler* para o *endpoint* equivalente no IP. O objetivo foi comparar códigos e estrutura de resposta em relação ao acesso via domínio.

Procedimento (*login*).

1. Criação de uma requisição `POST` para `https://<IP>:<PORT>/api/auth/login`.
2. Definição do cabeçalho `Content-Type: application/json`.
3. Uso do corpo JSON mapeado:
   `{"email":"usuario_autorizado@example.com","password":"SENHA_"}`
4. Tratamento de TLS/SNI ao utilizar IP em HTTPS (desativação de verificação no *Postman* apenas no ambiente de teste, ou mapeamento de *hostname* em `/etc/hosts`).
5. Envio de uma tentativa de controle e registro de *status*, tempo de resposta e trecho de mensagem.

Equivalente em `curl` (reprodutibilidade).

```
curl -k -X POST "https://<IP>:<PORT>/api/auth/login" \
    -H "Content-Type: application/json" \
    -d '{"email":"usuario_autorizado@example.com","password":"SENHA_"}'
    -i --max-time 10
```

(`-k` apenas em ambiente de teste para ignorar verificação de certificado quando usando IP direto em HTTPS[11]).

Interpretação dos resultados.

- **200/201**: autenticação bem-sucedida (fora do escopo deste passo).
- **401/403**: credenciais inválidas ou acesso negado (comportamento esperado).
- **404**: caminho inexistente para o alvo/rota testados.
- **429**: proteção de *rate limiting* em ação.
- **495/525/erros TLS**: incompatibilidade de TLS/SNI/certificado ao usar endereço IP.

A estrutura da resposta (formato JSON e campos) e os trechos de mensagem foram confrontados com os observados via domínio (Passo 1). Divergências poderiam indicar camadas distintas (CDN/WAF na borda versus servidor de origem) ou políticas diferentes.

A Figura 6 mostra o uso do *Postman* como cliente HTTP para validar o endereço IP candidato identificado nas etapas anteriores. A requisição `POST` reproduz exatamente o corpo JSON e os cabeçalhos mapeados no *Fiddler*, incluindo o `Content-Type: application/json`, mas agora enviados diretamente para `https://<IP>:<PORT>/api/auth/login`. Ao comparar o código de resposta, o tempo de retorno e os campos da mensagem com aqueles obtidos via domínio original, é possível verificar se o IP testado corresponde ao servidor de aplicação real ou se ainda há alguma camada intermediária (por exemplo, CDN/WAF) alterando o comportamento observado.

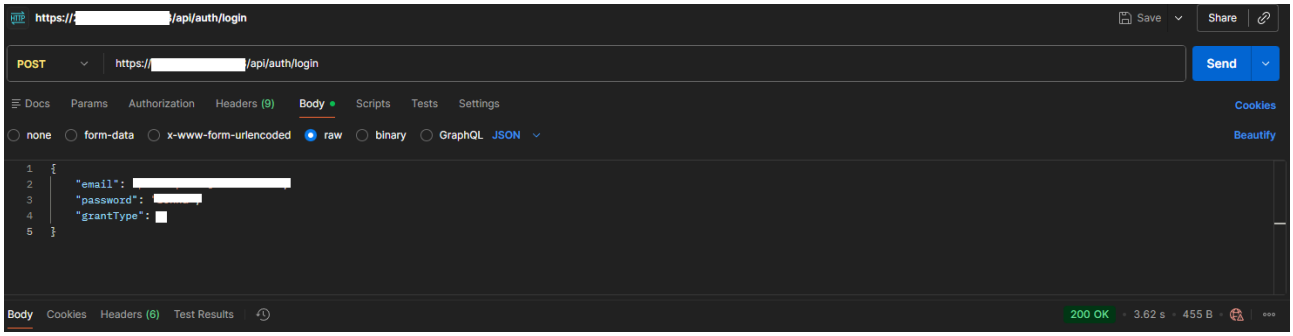

Figura 6. *Postman* — requisição `POST` para `https://<IP>:<PORT>/api/auth/login` com `Content-Type: application/json` e corpo equivalente ao mapeado no Fiddler.

5.4. Passo 4 — Obtenção de identificadores por OSINT (*Anymailfinder* e alternativas)

O objetivo desta etapa consistiu em levantar identificadores de conta (*e-mails* corporativos) associados ao domínio de interesse por meio de fontes abertas. Para acelerar a coleta, empregou-se a ferramenta *on-line Anymailfinder* [1], e registraram-se alternativas equivalentes presentes no *Kali Linux* que poderiam reproduzir o resultado com maior tempo de busca e consolidação.

Procedimento (*Anymailfinder*).

1. Consulta ao domínio corporativo na interface do serviço (`<dominio>`).

2. Coleta dos *e-mails* retornados e respectivos metadados (formato, grau de confiança, fonte).
3. Filtragem por padrão institucional (ex.: `nome.sobrenome@<dominio>`) e descarte de duplicatas/variações evidentes.
4. Registro dos candidatos válidos no inventário de OSINT, com anotação da fonte e data.

Alternativas nativas do *Kali* (resultado equivalente, maior esforço).

Sobre o "maior esforço". No contexto das alternativas nativas do *Kali*, "maior esforço" refere-se ao trabalho adicional de *preparação*, *orquestração* e *pós-processamento* que, em serviços on-line, costuma vir embutido. Na prática, isso implica: instalar/atualizar utilitários e, quando necessário, obter e gerir *API keys*; ajustar fontes e parâmetros (motores de busca, CT logs) para contornar *rate limits*/CAPTCHAs; normalizar e deduplicar resultados reduzindo *noise* (aliases, endereços genéricos, domínios *catch-all*); validar passivamente os achados (páginas e documentos públicos) sem contatar a aplicação-alvo; e registrar comandos e artefatos para permitir reprodutibilidade. Em síntese, obtém-se resultado equivalente ao do *Anymailfinder*, porém com maior investimento de tempo operacional e disciplina de higiene/validação dos dados coletados.

As ferramentas para isso incluem:

- `theHarvester` [8]: agregação de *e-mails* a partir de múltiplas fontes públicas (buscadores, redes, PGP etc.), com necessidade de *tuning* de chaves/fonte e posterior deduplicação.
- `amass` [24]/`subfinder` [30] + `ctfr`/`crt.sh`: enumeração de subdomínios e consulta a *Certificate Transparency*, útil para inferir padrões de nomenclatura e localizar páginas de contato/listas públicas.
- `holehe`: verificação passiva da existência de *e-mails* em provedores populares, apoiando validação sem envio de tráfego para o alvo. [19]

Validação e higienização.

1. Verificação de formato e domínio (RFC/consistência de MX).
2. Conferência passiva de existência (páginas institucionais, perfis públicos, menções em documentos) sem contato direto com a aplicação alvo.
3. Seleção de um único identificador autorizado para os testes subsequentes (controle do escopo e redução de impacto).

Limitações e vieses.

- **Cobertura de fontes**: serviços on-line variam em base e frequência de atualização; resultados refletem o recorte do período consultado.
- **Falsos positivos/negativos**: homônimos, redirecionamentos e *aliases* podem distorcer a contagem; a validação passiva reduz, mas não elimina, o risco.
- **Termos de uso e privacidade**: a coleta respeitou o caráter público das fontes e não envolveu técnicas de *scraping* intrusivo ou envio de mensagens.

A Figura 7 ilustra o uso de uma plataforma de descoberta de e-mails corporativos como fonte de *OSINT*. No painel à esquerda, é realizada uma busca do tipo *company search*, em que o domínio da organização é informado para iniciar a coleta. Já o painel à direita exibe o histórico de buscas e a lista de endereços encontrados, indicando a quantidade de e-mails válidos associados ao domínio e os diferentes formatos observados (por exemplo, combinações de nome e sobrenome). A partir desses resultados, o analista consegue inferir o padrão de formação de e-mails adotado pela empresa, reduzindo o espaço de busca para tentativas de autenticação indevida ou campanhas de *phishing*.

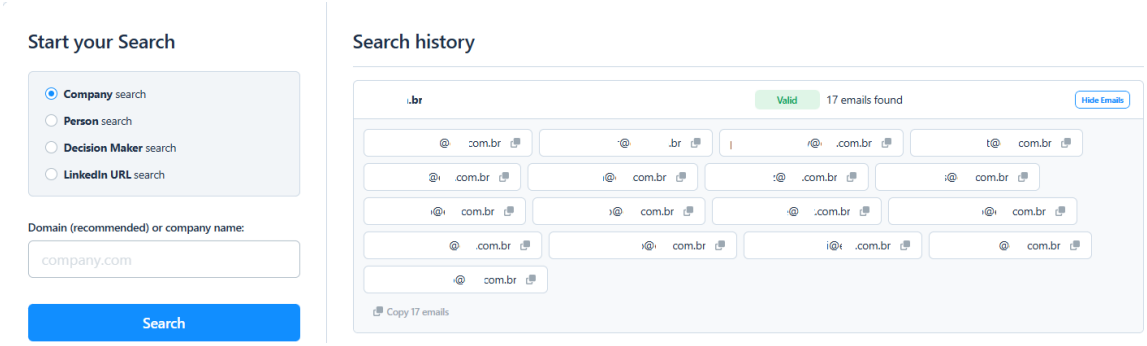

Figura 7. Exemplo de consulta OSINT para extração de padrões de e-mail no domínio de interesse.

5.5. Passo 5 — Teste controlado no endpoint de recuperação (mensagens e limites)

O levantamento por OSINT resultou em 17 *e-mails* corporativos candidatos. Para fins de estudo e conformidade ética, os testes práticos foram realizados apenas com dois identificadores: (i) um *e-mail* autorizado; e (ii) um *e-mail* inexistente (controle). A escolha recaiu sobre o *endpoint* de recuperação de senha, cuja granularidade de mensagens de erro se mostra elevada na evidência visual anexada, o que favorece a análise de consistência e a detecção de enumeração.

Objetivo.
Comparar respostas do endpoint `/api/auth/recovery-password` para um identificador válido versus um inexistente, medindo: código HTTP, latência, trecho de mensagem retornada e eventual ativação de *rate limiting*.

Procedimento.

1. Seleção dos identificadores: `email_autorizado@dominio` e `email_inexistente@dominio`.
2. Envio de requisições `POST` para `/api/auth/recovery-password` com `Content-Type: application/json` e corpo `{"email": "<valor>"}`.
3. Aplicação de limite de taxa (por exemplo, 20 tentativas/min) e janela curta de observação.
4. Registro telemétrico por tentativa: timestamp, código HTTP, latência (ms) e trecho de mensagem.
5. Consolidação dos resultados em CSV para comparação lado a lado.

Interpretação dos resultados.

- **Consistência de mensagem**: respostas indistinguíveis entre "válido" e "inexistente" indicam boa prática (reduz enumeração). Mensagens distintas ou códigos/latências divergentes reforçam risco de enumeração.
- ***Rate limiting*/bloqueio**: presença de `429` ou atrasos progressivos indica proteção ativa [27]. A ausência de reação sob sequência curta não invalida a existência de limiares mais altos.
- **Tempo/esforço**: número de tentativas até a primeira reação defensiva e o tempo total compõem o esforço prático observado.

Escopo e salvaguardas.
Os 17 e-mails obtidos por OSINT não foram utilizados em tentativas repetidas. O estudo restringiu

execuções ao e-mail autorizado e a um inexistente *apenas* para observar consistência de mensagens e a existência de limites básicos, sob taxa controlada e janelas curtas.

A Figura 8 ilustra o teste controlado realizado sobre o endpoint de recuperação de senha `/api/auth/recovery-password`. Na parte superior da captura, observa-se a requisição HTTP contendo apenas o campo `"email"` no corpo JSON. Na parte inferior, são exibidas as respostas retornadas pela aplicação para diferentes cenários (endereço cadastrado e endereço inexistente), com variações tanto no conteúdo da mensagem quanto nos campos estruturados da resposta. Essa diferença de granularidade fornece ao atacante um canal de enumeração de contas, permitindo inferir se um determinado e-mail está ou não registrado no sistema a partir do texto de erro e dos códigos retornados.

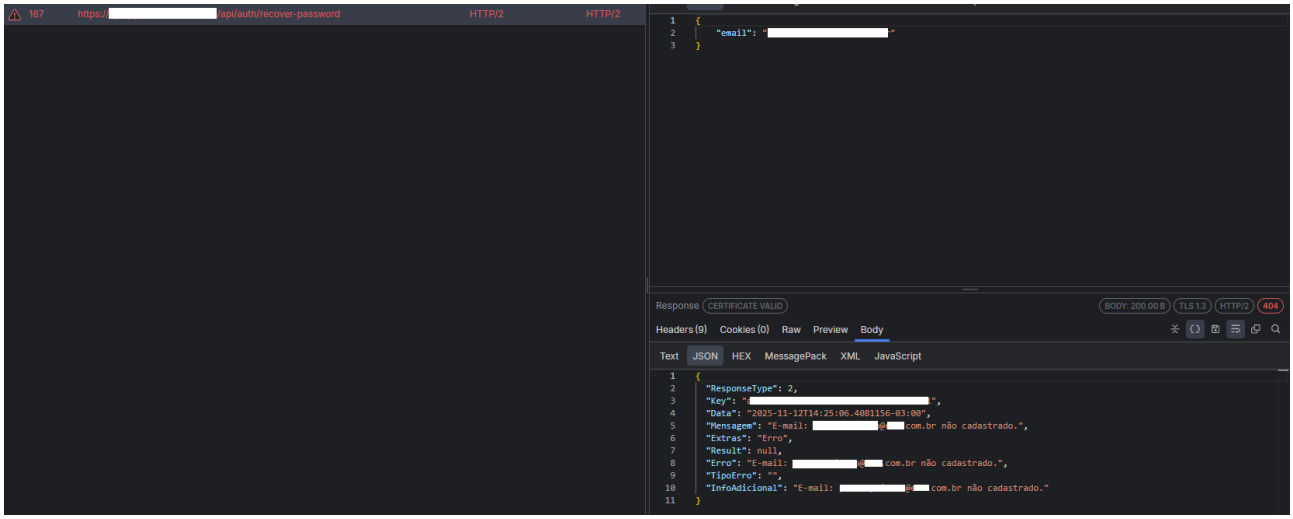

Figura 8. Evidência de granularidade de mensagens no endpoint `/api/auth/recovery-password` (exemplo sanitizado).

Resultados resumidos.
A Tabela 1 sintetiza os códigos de status HTTP observados no endpoint `/api/auth/recovery-password` para cada tipo de identificador testado. Nos experimentos, requisições com e-mail válido (autorizado) retornaram `200`, enquanto o e-mail inexistente (controle) retornou `401`. Essa distinção clara de códigos de resposta reforça o potencial de enumeração de contas, pois permite inferir a existência ou não de um identificador apenas a partir do status HTTP.

Tabela 1. Endpoint `/api/auth/recovery-password`: status por tipo de identificador.

Identificador	Status observado
Válido (autorizado)	200
Inexistente (controle)	401

5.6. Passo 6 — Tentativas controladas no *login* com *script* fixo (concorrência por *threads*)

Com um identificador autorizado em mãos, as tentativas *online* controladas no endpoint de *login* foram executadas por um script em *Python* simples, sem interface de linha de comando. Os valores de configuração permaneceram fixos no código-fonte: (i) URL do *endpoint* de *login*; (ii) *e-mail* autorizado; (iii) caminho da *wordlist*; e (iv) número de *threads* para concorrência.

Configuração embutida no script.
O *script* utilizou as seguintes constantes (exemplo genérico):

- `URL` → `https://<host>/api/auth/login`;
- `EMAIL` → `usuario_autorizado@dominio`;
- `WORDLIST` → `wordlist_8digitos.txt`;
- `THREADS` → `<valor>` (grau de concorrência);
- `OUT_CSV` → `login_results.csv`.

Wordlist de 8 dígitos.
A *wordlist* numérica foi produzida com o `crunch` [10] (comprimento fixo 8), e utilizada em amostras controladas:

```
crunch 8 8 0123456789 -o wordlist_8digitos.txt
```

Execução.
A execução foi direta, sem argumentos:

```
python3 login_bruteforce.py
```

Cada *thread* enviou requisições `POST` para `/api/auth/login` com o e-mail autorizado e senhas da *wordlist*, registrando por linha: *timestamp*, número da tentativa, código HTTP, latência e um trecho da mensagem de resposta no arquivo `login_results.csv`. A taxa efetiva resultou da combinação entre o número de `THREADS` e a latência média do alvo.

Telemetria e critérios.
Os dados coletados foram analisados para (i) detecção de `429` ou mensagens de bloqueio/atraso; (ii) consistência das mensagens de falha; e (iii) estimativa do esforço prático até a primeira reação defensiva.

Salvaguardas.
As tentativas permaneceram restritas ao *e-mail* autorizado, com janelas curtas e concorrência limitada por `THREADS`. Não houve execução exaustiva da *wordlist*; empregaram-se amostras para caracterização de controles.

A Figura 9 apresenta o fluxograma do experimento de autenticação com múltiplas senhas de teste em ambiente controlado. O processo inicia com a definição das variáveis fixas (endpoint, cabeçalhos, e-mail autorizado, número de *threads* e limites de requisições). Em seguida, ocorre a leitura da *wordlist* contendo as senhas candidatas e sua distribuição entre as *threads* de execução.

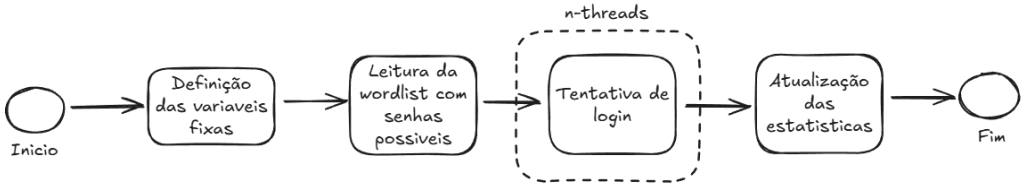

Figura 9. Fluxograma do experimento de autenticação com múltiplas senhas de teste em ambiente controlado.

Cada *thread* realiza tentativas de *login* de forma concorrente, registrando código de status, tempo de resposta e demais campos relevantes. Por fim, as estatísticas são atualizadas e consolidadas,

produzindo um resumo quantitativo do experimento para análise posterior.

Saída consolidada do terminal.
A Listagem 1 apresenta o resumo final produzido pelo *script*, indicando o caminho do artefato salvo no disco e a distribuição dos códigos HTTP observados na amostra. O bloco mostra, primeiro, a contagem bruta de respostas retornadas pelo endpoint e, em seguida, o mesmo resultado agregado por categoria (sucesso, credenciais inválidas, proteção por *rate limiting*, erros de servidor e outros), o que facilita a interpretação quantitativa do experimento.

Listing 1. Resumo final das tentativas no login (saída do terminal).

```
Resultado salvo em: /home/kali/Documents/brute_force/
   resultado_credenciais.csv

Resumo de codigos HTTP observados (bruto):
  200: 1 vez(es)
  401: 1259 vez(es)

Resumo agregado por categoria:
  200 (sucesso): 1
  401/403 (credenciais invalidas): 1259
  429 (rate limit / protecao): 0
  5xx (erros de servidor): 0
  Outros: 0
```

Síntese do passo.
A amostra executada evidenciou predominância de falhas de credenciais (`401`) e ausência de reação explícita de *rate limiting* (`429`) no intervalo observado. O único retorno `200` caracteriza caso isolado na amostra, insuficiente para inferir comprometimento.

Os resultados reforçam a necessidade de controles complementares no ponto de autenticação, tais como mensagens de erro uniformes, limites por conta/origem com janelas deslizantes, atrasos progressivos e MFA com verificação contra listas de senhas comprometidas [27, 13].

6. Conclusão, Soluções e Trabalhos Futuros

O estudo evidenciou que fontes abertas (OSINT) reduzem substancialmente o espaço de busca na superfície de autenticação de aplicações *SaaS*, sobretudo quando há exposição de padrões institucionais de *e-mail* e quando mensagens/códigos de erro diferem entre casos válidos e inexistentes.

A experimentação controlada — limitada a um identificador autorizado e a um inexistente — mostrou predominância de respostas `401` e ausência de *rate limiting* explícito na amostra, além de comportamento distinto no fluxo de recuperação, suficiente para caracterizar risco de enumeração por resposta. A metodologia preservou o escopo ético, permitiu mensurar esforço temporal e forneceu evidências objetivas para orientar endurecimento no ponto de *login*, em linha com diretrizes nacionais (ANPD, CMN/BCB) e guias boas práticas internacionais (OWASP/NIST) [2, 9, 25, 13].

A Tabela 2 relaciona achados a recomendações técnicas e métricas operacionais. As medidas seguem referências brasileiras — como a LGPD e guias da ANPD — e padrões técnicos amplamente aceitos (OWASP/NIST). Para instituições financeiras e de pagamento, a política de segurança cibernética e os requisitos de contratação de serviços em nuvem previstos pela regulação do CMN/BCB também são pertinentes [5, 2, 9].

Tabela 2. Achados, mitigações recomendadas e métricas de acompanhamento (síntese).

Achado	Mitigação recomendada	Métrica/indicador
Enumeração por resposta (mensagens/códigos distintos)	Mensagens *uniformes* de falha no *login* e na recuperação; padronização entre fluxos; ocultar a razão específica ao usuário.	% de respostas indistinguíveis; variação de latência entre casos (ms).
Ausência de *rate limiting* efetivo	Limites por conta e por origem (IP/ASN), janela deslizante, atraso progressivo e bloqueio temporário; exceções orientadas por risco.	Tempo até o 1º `429`; tentativas até bloqueio; incidência de `429`/sessão.
Higiene de exposição pública (sites, docs, CT logs) e padrão previsível de e-mail	Redução de exposição de identificadores; e-mails públicos menos previsíveis; convites *out-of-band* para cadastros sensíveis.	Queda na taxa de acerto em enumeração controlada (OSINT).
Credenciais fracas/reutilizadas	Verificação contra listas comprometidas; política de senha resistente a adivinhação; MFA orientada por risco (dispositivo/geo/comportamento).	% de senhas bloqueadas; adoção de MFA; detecções de *stuffing*.
Baixa visibilidade operacional	Telemetria estruturada no *login* (código, latência, fingerprint leve, ASN), alertas por *burst*, *dashboards* de abuso.	MTTD; bursts/hora por ASN; taxa de falsos positivos.

Recomenda-se, adicionalmente: (i) controles no fluxo de recuperação (prova de posse do canal, *rate limit* dedicado, mensagens uniformes); (ii) fricção adaptativa sob sinais de risco; (iii) revisão de exposição pública (páginas, repositórios, *sitemaps*, documentos); (iv) programa contínuo com testes canário e SLIs/SLOs de autenticação segura [2, 7, 25].

Trabalhos futuros.
Linhas de continuidade incluem: (i) avaliação longitudinal da efetividade das mitigações com séries temporais de enumeração controlada, incidência de `429`, tempo até bloqueio e adoção de MFA; (ii) ampliação para *password spraying* e *credential stuffing* com telemetria comportamental[27]; (iii) comparação entre camadas de proteção (WAF/CDN, *bot management*, desafios graduais) e políticas de autenticação (MFA condicional, listas comprometidas, sessões resistentes a *replay*); (iv) estudo de *honeypots* de autenticação; e (v) painéis operacionais dedicados ao abuso, integrando dados de gateway, aplicação e IdP para reduzir MTTD/MTTR [25, 13, 2].

Referências

[1] Anymailfinder. Anymailfinder. `https://anymailfinder.com/`. Serviço de descoberta de e-mails corporativos. Acesso em: nov. 2025.

[2] Autoridade Nacional de Proteção de Dados (ANPD). Guia orientativo sobre segurança da informação para agentes de tratamento de pequeno porte. `https://www.gov.br/anpd/pt-br/centrais-de-conteudo/materiais-educativos-e-publicacoes/guia-vf.pdf`, 2021. Acesso em: nov. 2025.

[3] BinaryEdge. Binaryedge. `https://www.binaryedge.io/`, 2014. Plataforma de varredura e inteligência de exposição em larga escala na Internet. Acesso em: nov. 2025.

[4] Joseph Bonneau, Cormac Herley, Paul C. van Oorschot, and Frank Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *2012*

IEEE Symposium on Security and Privacy (SP), pages 553–567. IEEE, 2012.

[5] Brasil. Lei geral de proteção de dados pessoais (lgpd) — lei nº 13.709, de 14 de agosto de 2018. https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm, 2018. Presidência da República. Acesso em: nov. 2025.

[6] Censys, Inc. Censys search. https://search.censys.io/. Plataforma de levantamento de ativos expostos. Acesso em: nov. 2025.

[7] CERT.br/NIC.br. Cartilha de segurança para internet (versão atual). https://cartilha.cert.br/, 2024. Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil. Acesso em: nov. 2025.

[8] Christian Martorella e colaboradores. theharvester. https://github.com/laramies/theHarvester. Ferramenta OSINT para coleta de e-mails/hosts. Acesso em: nov. 2025.

[9] CMN/BCB. Resolução cmn nº 4.893, de 26 de fevereiro de 2021 — política de segurança cibernética e requisitos para contratação de serviços em nuvem, 2021. Conselho Monetário Nacional / Banco Central do Brasil. Acesso em: nov. 2025.

[10] Crunch Project. crunch password list generator. https://sourceforge.net/projects/crunch-wordlist/. Ferramenta para geração de wordlists. Acesso em: nov. 2025.

[11] Daniel Stenberg e colaboradores. curl. https://curl.se/. Ferramenta de linha de comando para transferência de dados. Acesso em: nov. 2025.

[12] GNU Project. *GNU grep Manual*. Free Software Foundation, 2024. Acesso em: nov. 2025.

[13] Paul A. Grassi, Michael E. Garcia, and James L. Fenton. Digital identity guidelines: Authentication and lifecycle management. Technical Report Special Publication 800-63B, NIST, 2017. Inclui atualizações revisadas em 2020+.

[14] HTTPie Project. Httpie. https://httpie.io/. Cliente HTTP de linha de comando. Acesso em: nov. 2025.

[15] Knownsec Inc. Zoomeye. https://www.zoomeye.org/, 2013. Motor de busca em ativos de redes e serviços expostos na Internet. Acesso em: nov. 2025.

[16] jq Project. jq. https://jqlang.github.io/jq/. Processador de JSON em linha de comando. Acesso em: nov. 2025.

[17] Gordon Lyon. Nmap network mapper. https://nmap.org/. Ferramenta para varredura e auditoria de redes. Acesso em: nov. 2025.

[18] Macêdo, A. S. Framework de inteligencia cibernetica para interacao humana utilizando as melhores praticas de fontes abertas. Technical report, Departamento de Engenharia Eletrica, Universidade de Brasilia, Brasília, DF, 2023. Dissertacao de Mestrado Profissional.

[19] megadose. Holehe. https://github.com/megadose/holehe, 2020. Ferramenta OSINT para verificar se um e-mail está vinculado a contas em múltiplas plataformas. Acesso em: nov. 2025.

[20] mitmproxy Project. mitmproxy. https://mitmproxy.org/. Intercepting proxy HTTP(S) de código aberto. Acesso em: nov. 2025.

[21] NATO. Nato open source intelligence handbook, 2001. Definições e princípios clássicos de OSINT.

[22] Offensive Security. Kali linux. https://www.kali.org/. Acesso em: nov. 2025.

[23] Oracle Corporation. Oracle vm virtualbox. https://www.virtualbox.org/. Acesso em: nov. 2025.

[24] OWASP. Owasp amass project. https://github.com/owasp-amass/amass. Enumeração de superfície externa. Acesso em: nov. 2025.

[25] OWASP. Owasp application security verification standard 4.x, 2019. Guia de requisitos de segurança de aplicação.

[26] OWASP. Owasp top 10: 2021, 2021. Riscos mais críticos em aplicações web.

[27] OWASP. Authentication cheat sheet, 2023. Boas práticas de autenticação.

[28] PortSwigger Ltd. Burp suite. https://portswigger.net/burp. Plataforma integrada para testes de segurança web. Acesso em: nov. 2025.

[29] Postman, Inc. Postman. https://www.postman.com/. Plataforma para desenvolvimento e teste de APIs. Acesso em: nov. 2025.

[30] ProjectDiscovery. subfinder. https://github.com/projectdiscovery/subfinder. Ferramenta para enumeração de subdomínios. Acesso em: nov. 2025.

[31] Python Software Foundation. Python 3. https://www.python.org/. Linguagem de programação Python. Acesso em: nov. 2025.

[32] Kenneth Reitz and colaboradores. Requests: Http for humans. https://requests.readthedocs.io/. Biblioteca HTTP para Python. Acesso em: nov. 2025.

[33] Arnold D. Robbins. *GAWK: Effective AWK Programming — A User's Guide for GNU Awk*. Free Software Foundation, 2023. Acesso em: nov. 2025.

[34] Shodan. Shodan. https://www.shodan.io/, 2009. Motor de busca em dispositivos conectados à Internet. Acesso em: nov. 2025.

[35] Telerik. Fiddler classic / fiddler everywhere. https://www.telerik.com/fiddler. Proxy HTTP(S) para depuração. Acesso em: nov. 2025.