



OWASP API Security Top 10 2019

Os dez riscos de segurança de API mais críticos

Sumário do

Índice

Sumário do TOC.....	2
Prefácio.....	3
Introdução	4
Notas de versão do RN.....	5
RISK API Security	
Risk.....	6
T10 OWASP API Security Top 10 - 2019..	7
API1:2019 Autorização de nível de objeto quebrada.....	8
API2:2019 Autenticação de usuário quebrada.....	10
API3:2019 Exposição Excessiva de Dados.....	12
API4:2019 Falta de Recursos e Limitação de Taxa.	14
API5:2019 Autorização de nível de função quebrada...16	
API6:2019 Atribuição em massa.....	18
API7:2019 Configuração incorreta de segurança.....	20
API8:2019 Injeção.....	22
API9:2019 Gerenciamento de ativos impróprios.....	24
API10:2019 Registro e monitoramento insuficientes.....	26
+D Wha O que vem a seguir para os desenvolvedores.....	28
+DSO O que vem a seguir para DevSecOps.....	29
Metodologia e Dados +DAT.....	30
+ACK Agradecimentos.....	31

Sobre a OWASP

O Open Web Application Security Project (OWASP) é uma comunidade aberta dedicada a permitir que as organizações desenvolvam, adquiram e mantenham aplicativos e APIs confiáveis.

Na OWASP, você encontrará gratuitamente e aberto:

- Ferramentas e padrões de segurança de aplicativos. • Livros completos sobre segurança de aplicativos testes, desenvolvimento de código seguro e revisão de código seguro. • Apresentações e vídeos. • Folhas de dicas em muitos tópicos comuns. • Controles e bibliotecas de segurança padrão. • Capítulos locais em todo o mundo. • Pesquisa inovadora. • Extensas conferências em todo o mundo.

- [Listas de discussão.](#)

Saiba mais em: <https://www.owasp.org>.

Todas as ferramentas, documentos, vídeos, apresentações e capítulos do OWASP são gratuitos e abertos a qualquer pessoa interessada em melhorar a segurança do aplicativo.

Defendemos abordar a segurança de aplicativos como um problema de pessoas, processos e tecnologia porque as abordagens mais eficazes para a segurança de aplicativos exigem melhorias nessas áreas.

OWASP é um novo tipo de organização. Nossa liberdade de pressões comerciais nos permite fornecer informações imparciais, práticas e econômicas sobre segurança de aplicativos.

A OWASP não é afiliada a nenhuma empresa de tecnologia, embora apoiemos o uso informado da tecnologia de segurança comercial. A OWASP produz diversos tipos de materiais de forma colaborativa, transparente e aberta.

A Fundação OWASP é a entidade sem fins lucrativos que garante o sucesso do projeto a longo prazo. Quase todos os associados à OWASP são voluntários, incluindo o conselho da OWASP, líderes de capítulo, líderes de projeto e membros do projeto.

Apoiamos pesquisas inovadoras de segurança com subsídios e infraestrutura.

Junte-se a nós!

FW Prefácio

Um elemento fundamental da inovação no mundo orientado a aplicativos de hoje é a interface de programação de aplicativos (API). De bancos, varejo e transporte a IoT, veículos autônomos e cidades inteligentes, as APIs são uma parte crítica dos aplicativos móveis, SaaS e da Web modernos e podem ser encontradas em aplicativos voltados para o cliente, voltados para parceiros e internos.

Por natureza, as APIs expõem a lógica do aplicativo e dados confidenciais, como informações de identificação pessoal (PII) e, por causa disso, as APIs se tornaram cada vez mais um alvo para invasores. Sem APIs seguras, a inovação rápida seria impossível.

Embora um Top 10 mais amplo dos riscos de segurança de aplicativos da Web ainda faça sentido, devido à sua natureza específica, é necessária uma lista de riscos de segurança específica da API. A segurança da API concentra-se em estratégias e soluções para entender e mitigar as vulnerabilidades exclusivas e os riscos de segurança associados às APIs.

Se você está familiarizado com o [projeto OWASP Top 10](#), então você notará as semelhanças entre os dois documentos: eles se destinam à legibilidade e adoção. Se você é novo na série OWASP Top 10, talvez seja melhor ler as seções [API Security Risks](#) and [Methodology and Data](#) antes de pular para a [lista Top 10](#).

Você pode contribuir para o OWASP API Security Top 10 com suas perguntas, comentários e ideias em nosso repositório de projetos GitHub:

- <https://github.com/OWASP/API-Security/issues> • <https://github.com/OWASP/API-Security/blob/master/CONTRIBUTING.md>

Você pode encontrar o OWASP API Security Top 10 aqui:

- https://www.owasp.org/index.php/OWASP_API_Security_Project • <https://github.com/OWASP/API-Security>

Gostaríamos de agradecer a todos os colaboradores que tornaram este projeto possível com seus esforços e contribuições. Eles estão todos listados na [seção Agradecimentos](#). Obrigado!

Introdução

Bem-vindo ao OWASP API Security Top 10 - 2019!

Bem-vindo à primeira edição do OWASP API Security Top 10. Se você estiver familiarizado com a série OWASP Top 10, notará as semelhanças: eles são destinados à legibilidade e adoção. Caso contrário, considere visitar a [página wiki do OWASP API Security Project](#), antes de se aprofundar nos riscos de segurança de API mais críticos.

As APIs desempenham um papel muito importante na arquitetura dos aplicativos modernos. Como a conscientização e a inovação da segurança têm ritmos diferentes, é importante focar nas falhas comuns de segurança da API.

O principal objetivo do OWASP API Security Top 10 é educar os envolvidos no desenvolvimento e manutenção de API, por exemplo, desenvolvedores, designers, arquitetos, gerentes ou organizações.

Na seção [Metodologia e Dados](#), você pode ler mais sobre como essa primeira edição foi criada. Em versões futuras, queremos envolver a indústria de segurança, com uma chamada pública de dados. Por enquanto, encorajamos todos a contribuir com perguntas, comentários e ideias em nosso [repositório GitHub](#) ou [lista de discussão](#).

Notas de lançamento

Esta é a primeira edição do OWASP API Security Top 10, que planejamos ser atualizada periodicamente, a cada três ou quatro anos.

Ao contrário desta versão, em versões futuras, queremos fazer uma chamada pública de dados, envolvendo a indústria de segurança neste esforço. Na seção [Metodologia e Dados](#), você encontrará mais detalhes sobre como esta versão foi construída. Para obter mais detalhes sobre os riscos de segurança, consulte a seção [API Security Risks](#).

É importante perceber que, nos últimos anos, a arquitetura dos aplicativos mudou significativamente.

Atualmente, as APIs desempenham um papel muito importante nessa nova arquitetura de microsserviços, Single Page Applications (SPAs), aplicativos móveis, IoT, etc.

O OWASP API Security Top 10 foi um esforço necessário para criar consciência sobre os problemas de segurança da API moderna. Isso só foi possível graças ao grande esforço de vários voluntários, todos listados na seção [Agradecimentos](#).
Obrigado!

RISCO Risco de segurança da API

A [Metodologia de Classificação de Risco OWASP](#) foi usado para fazer a análise de risco.

A tabela abaixo resume a terminologia associada à pontuação de risco.

Exploração de agentes de ameaças		Fraqueza Prevalência	Fraqueza Detectabilidade	Técnico Impacto	O negócio Impactos
Específico da API	Fácil: 3	generalizado 3	Fácil 3	Grave 3	O negócio Específico
	Média: 2	Comum 2	Média 2	Moderado 2	
	Difícil: 1	Difícil 1	Difícil 1	Menor 1	

Observação: essa abordagem não leva em consideração a probabilidade do agente de ameaça. Também não é responsável por nenhum dos vários detalhes técnicos associados ao seu aplicativo específico. Qualquer um desses fatores pode afetar significativamente a probabilidade geral de um invasor encontrar e explorar uma vulnerabilidade específica. Essa classificação não leva em consideração o impacto real no seu negócio. Sua organização terá que decidir quanto risco de segurança de aplicativos e APIs a organização está disposta a aceitar de acordo com sua cultura, setor e ambiente regulatório. O objetivo do OWASP API Security Top 10 não é fazer essa análise de risco para você.

Referências

OWASP

- [Metodologia de Classificação de Risco OWASP](#)
- [Artigo sobre Modelagem de Ameaças/Riscos](#)


Externo

- [ISO 31000: Norma de Gestão de Risco](#)
- [ISO 27001: ISMS](#)
- [NIST Cyber Framework \(EUA\)](#) • [ASD Mitigações Estratégicas \(AU\)](#) • [NIST CVSS 3.0](#)
- [Ferramenta de Modelagem de Ameaças da Microsoft](#)

T10 OWASP API Security Top 10 - 2019

API1:2019 - APIs de autorização de nível de objeto quebrado tendem a	expor pontos de extremidade que lidam com identificadores de objeto, criando um problema de controle de acesso de nível de superfície de ataque amplo. As verificações de autorização em nível de objeto devem ser consideradas em todas as funções que acessam uma fonte de dados usando uma entrada do usuário.
API2:2019 - Autenticação de usuário quebrada	Os mecanismos de autenticação geralmente são implementados incorretamente, permitindo que os invasores comprometam os tokens de autenticação ou explorem falhas de implementação para assumir as identidades de outros usuários temporária ou permanentemente. Comprometer a capacidade do sistema de identificar o cliente/usuário compromete a segurança geral da API.
API3:2019 - Exposição excessiva de dados	Ansiosos por implementações genéricas, os desenvolvedores tendem a expor todas as propriedades do objeto sem considerar sua sensibilidade individual, contando com os clientes para realizar a filtragem dos dados antes de exibi-los ao usuário.
API4:2019 - Falta de recursos e limitação de taxa Muitas vezes, as APIs	não impõem nenhuma restrição quanto ao tamanho ou número de recursos que podem ser solicitados pelo cliente/usuário. Isso não apenas pode afetar o desempenho do servidor API, levando à negação de serviço (DoS), mas também deixa a porta aberta para falhas de autenticação, como força bruta.
API5:2019 - Nível de função quebrada Autorização	Políticas complexas de controle de acesso com diferentes hierarquias, grupos e papéis, e uma separação pouco clara entre funções administrativas e regulares, tendem a levar a falhas de autorização. Ao explorar esses problemas, os invasores obtêm acesso aos recursos e/ou funções administrativas de outros usuários.
API6:2019 - Atribuição em massa	A vinculação de dados fornecidos pelo cliente (por exemplo, JSON) a modelos de dados, sem a filtragem adequada de propriedades com base em uma lista de permissões, geralmente leva à atribuição em massa. Adivinhar as propriedades dos objetos, explorar outros endpoints da API, ler a documentação ou fornecer propriedades adicionais do objeto em payloads de solicitação permite que os invasores modifiquem as propriedades do objeto que não deveriam.
API7:2019 - Configuração incorreta de segurança	A configuração incorreta de segurança geralmente é resultado de configurações padrão inseguras, configurações incompletas ou ad-hoc, armazenamento em nuvem aberta, cabeçalhos HTTP mal configurados, métodos HTTP desnecessários, compartilhamento permissivo de recursos Cross-Origin (CORS) e mensagens de erro detalhadas contendo informações confidenciais.
API8:2019 - Injeção	Falhas de injeção, como SQL, NoSQL, injeção de comando, etc., ocorrem quando dados não confiáveis são enviados a um interpretador como parte de um comando ou consulta. Os dados mal-intencionados do invasor podem induzir o intérprete a executar comandos indesejados ou acessar dados sem a devida autorização.
API9:2019 - Gerenciamento inadequado de ativos	As APIs tendem a expor mais endpoints do que os aplicativos da Web tradicionais, tornando a documentação adequada e atualizada altamente importante. Os hosts adequados e o inventário de versões de API implantados também desempenham um papel importante para atenuar problemas como versões de API obsoletas e pontos de extremidade de depuração expostos.
API10:2019 - Registro e monitoramento insuficientes	Registro e monitoramento insuficientes, juntamente com integração ausente ou ineficaz com resposta a incidentes, permitem que os invasores ataquem ainda mais os sistemas, mantenham a persistência, mudem para mais sistemas para adulterar, extrair ou destruir dados. A maioria dos estudos de violação demonstra que o tempo para detectar uma violação é superior a 200 dias, geralmente detectado por partes externas, em vez de processos internos ou monitoramento.

API:2019 Autorização de nível de objeto quebrado

				
Exploração	específica da API : 3	Prevalência: 3	Detectabilidade: 2	Técnica: 3
Exploração	Os invasores podem explorar a API. Esse tem sido o acesso não autorizado a dados para manipulação de controle de acesso quebrado em partes não autorizadas modificando dados e publicação de dados.	A API não possui uma estrutura adequada de acesso não autorizado para verificação e essa autorização não pode ser desenvolvida com verificações de controle de dados. Esse problema é geralmente não resolvido automaticamente. O componente geralmente não é o único que está disponível para discar o objeto de	A API não possui uma estrutura adequada de acesso não autorizado para verificação e essa autorização não pode ser desenvolvida com verificações de controle de dados. Esse problema é geralmente não resolvido automaticamente. O componente geralmente não é o único que está disponível para discar o objeto de	Específicos de negócios Os invasores podem explorar a API. Esse tem sido o acesso não autorizado a dados para manipulação de controle de acesso quebrado em partes não autorizadas modificando dados e publicação de dados.

A API é vulnerável?

A autorização no nível do objeto é um mecanismo de controle de acesso que geralmente é implementado no nível do código para validar que um usuário só pode acessar os objetos aos quais ele deveria ter acesso.

Cada terminal de API que recebe um ID de um objeto e executa qualquer tipo de ação no objeto deve implementar verificações de autorização no nível do objeto. As verificações devem validar se o usuário conectado tem acesso para executar a ação solicitada no objeto solicitado.

Falhas nesse mecanismo geralmente levam à divulgação não autorizada de informações, modificação ou destruição de todos os dados.

Exemplos de Cenários de Ataque

Cenário 1

Uma plataforma de comércio eletrônico para lojas online (lojas) fornece uma página de listagem com os gráficos de receita de suas lojas hospedadas. Ao inspecionar as solicitações do navegador, um invasor pode identificar os endpoints da API usados como fonte de dados para esses gráficos e seu padrão /shops/{shopName}/revenue_data.json. Usando outro terminal de API, o invasor pode obter a lista de todos os nomes de lojas hospedadas. Com um script simples para manipular os nomes da lista, substituindo {shopName} na URL, o invasor obtém acesso aos dados de vendas de milhares de e

lojas de comércio.

Cenário #2

Ao monitorar o tráfego de rede de um dispositivo vestível, a seguinte solicitação HTTP PATCH chama a atenção de um invasor devido à presença de um cabeçalho de solicitação HTTP personalizado X-User-Id: 54796.

Substituindo o valor X-User-Id por 54795, o invasor recebe uma resposta HTTP bem-sucedida e pode modificar os dados da conta de outros usuários.

API1:2019 Autorização de nível de objeto quebrado

Como prevenir

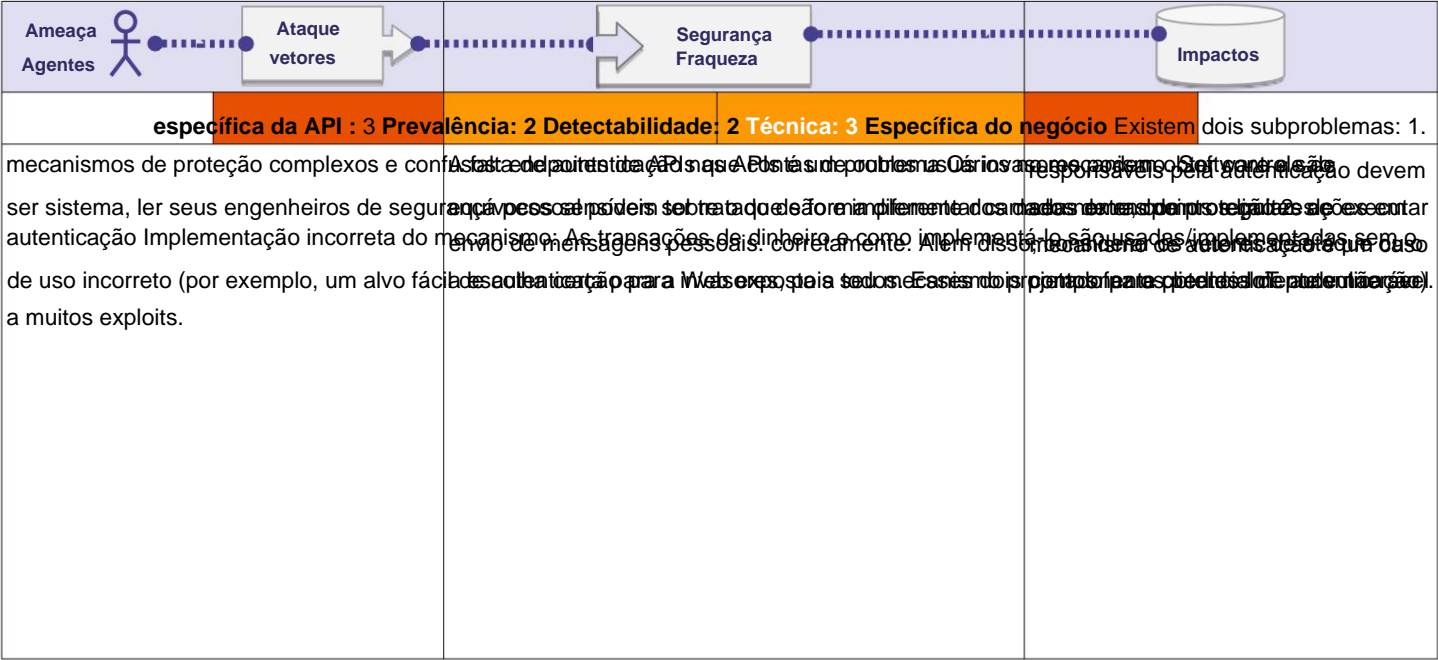
- Implemente um mecanismo de autorização adequado que se baseie nas políticas e na hierarquia do usuário. • Usar um mecanismo de autorização para verificar se o usuário logado tem acesso para realizar a ação solicitada no registro em cada função que utiliza uma entrada do cliente para acessar um registro no banco de dados.
- Prefira usar valores aleatórios e imprevisíveis como GUIDs para IDs de registros. • Escrever testes para avaliar o mecanismo de autorização. Não implemente mudanças vulneráveis que quebrem o testes.

Referências

Externo

- [CWE-284: Controle de acesso impróprio](#) •
- [CWE-285: Autorização imprópria](#) • [CWE-639: Desvio de autorização por meio de chave controlada pelo usuário](#)

API2:2019 Autenticação de usuário quebrada



A API é vulnerável?

Os endpoints e fluxos de autenticação são ativos que precisam ser protegidos. “Esqueceu a senha / redefinir senha” deve ser tratado da mesma forma que os mecanismos de autenticação.

Uma API é vulnerável se:

- Permite [preenchimento de credenciais](#) em que o invasor tem uma lista de nomes de usuário e senhas válidos.
- Permite que invasores executem um ataque de força bruta na mesma conta de usuário, sem apresentar mecanismo de bloqueio de captcha/conta.
- Permite senhas fracas.
- Envia detalhes de autenticação confidenciais, como tokens de autenticação e senhas no URL.
- Não valida a autenticidade dos tokens.
- Aceita tokens JWT não assinados/com assinatura fraca ("alg":"nenhum")/não valida sua data de validade.
- Usa senhas de texto simples, não criptografadas ou com hash fraco.
- Usa chaves de criptografia fracas.

Exemplos de Cenários de Ataque

Cenário 1

[Preenchimento de credenciais](#) (usando [listas de nomes de usuários/senhas conhecidos](#)), é um ataque comum. Se um aplicativo não implementar proteções automatizadas contra ameaças ou preenchimento de credenciais, o aplicativo poderá ser usado como um oráculo de senha (testador) para determinar se as credenciais são válidas.

Cenário #2

Um invasor inicia o fluxo de trabalho de recuperação de senha emitindo uma solicitação POST para / api/system/verification-codes e fornecendo o nome de usuário no corpo da solicitação. Em seguida, um token SMS com 6 dígitos é enviado para o telefone da vítima. Como a API não implementa uma política de limitação de taxa, o invasor pode testar todas as combinações possíveis usando um script multiencadeado, contra o endpoint /api/system/verification-codes/{smsToken} para descobrir o token certo em alguns minutos.

API2:2019 Autenticação de usuário quebrada

Como prevenir

- Certifique-se de conhecer todos os fluxos possíveis para autenticação na API (mobile/web/deep links que implementam autenticação com um clique/etc.)

- Pergunte a seus engenheiros quais fluxos você perdeu. •

Leia sobre seus mecanismos de autenticação. Certifique-se de entender o que e como eles são usados.

OAuth não é autenticação e nem chaves de API.

- Não reinvente a roda na autenticação, geração de tokens, armazenamento de senhas. Use os padrões. • Endpoints de recuperação/esquecimento de senha de credencial devem ser tratados como endpoints de login em termos de força bruta, limitação de taxa e proteções de bloqueio. • Use o [Cheatsheet de Autenticação OWASP](#). • Sempre que possível, implemente a autenticação multifator. • Implemente mecanismos anti-força bruta para mitigar o preenchimento de credenciais, ataque de dicionário e ataques de força bruta em seus endpoints de autenticação. Esse mecanismo deve ser mais rigoroso do que o mecanismo de limitação de taxa regular em sua API.

- Implementar [bloqueio de conta](#) / mecanismo captcha para evitar força bruta contra usuários específicos.

Implemente verificações de senha fraca.

- As chaves de API não devem ser usadas para autenticação de usuário, mas para [autenticação de aplicativo/projeto cliente](#).

Referências

OWASP

- [Folha de dicas de gerenciamento de chaves OWASP](#)
- [Cheatsheet de Autenticação OWASP](#)
- [Preenchimento de credenciais](#)

Externo

- [CWE-798: Uso de credenciais codificadas](#)

API3:2019 Exposição excessiva de dados

	Ameaça Agentes	Ataque vetores	Segurança Fraqueza	Impactos
Exploração	específica da API : 3	Prevalência: 2	Detectabilidade: 2	Exploração de Técnico: 2 A exposição excessiva de
dados excessivos APIs dependem de clientes para generalizar a exposição. Como as APIs são usadas como dados executados pelo cliente, os dados são enviados de volta ao cliente da API, implementá-las de maneira genérica simplifica a tarefa de exposição, não querendo ser exposta. As ferramentas automáticas geralmente não desse tipo de vulnerabilidade porque é difícil diferenciar entre dados legítimos e dados de API e dados confidenciais conhecimento profundo do aplicativo.	sites para generalizar a exposição. Como as APIs são usadas como dados executados pelo cliente, os dados são enviados de volta ao cliente da API, implementá-las de maneira genérica simplifica a tarefa de exposição, não querendo ser exposta. As ferramentas automáticas geralmente não desse tipo de vulnerabilidade porque é difícil diferenciar entre dados legítimos e dados de API e dados confidenciais conhecimento profundo do aplicativo.	dados específicos do negócio geralmente leva à exposição de dados confidenciais.		

A API é vulnerável?

A API retorna dados confidenciais para o cliente por design. Esses dados geralmente são filtrados no lado do cliente antes de serem apresentados ao usuário. Um invasor pode facilmente farejar o tráfego e ver os dados confidenciais.

Exemplos de Cenários de Ataque

Cenário 1

A equipe móvel usa o ponto de extremidade `/api/articles/{articleId}/comments/{commentId}` na exibição de artigos para renderizar metadados de comentários. Ao farejar o tráfego do aplicativo móvel, um invasor descobre que outros dados confidenciais relacionados ao autor do comentário também são retornados. A implementação do endpoint usa um método `toJSON()` genérico no modelo `User`, que contém PII, para serializar o objeto.

Cenário #2

Um sistema de vigilância baseado em IOT permite que os administradores criem usuários com diferentes permissões. Um administrador criou uma conta de usuário para um novo guarda de segurança que só deve ter acesso a edifícios específicos no local. Depois que o segurança usa seu aplicativo móvel, uma chamada de API é acionada para: `/api/sites/111/cameras` para receber dados sobre as câmeras disponíveis e mostrá-las no painel. A resposta contém uma lista com detalhes sobre câmeras no seguinte formato: `{"id":"xxx","live_access_token":"xxxx bbbbbb","building_id":"yyy"}`. Embora a GUI do cliente mostre apenas as câmeras às quais o guarda de segurança deve ter acesso, a resposta real da API contém uma lista completa de todas as câmeras no site.

API3:2019 Exposição excessiva de dados

Como prevenir

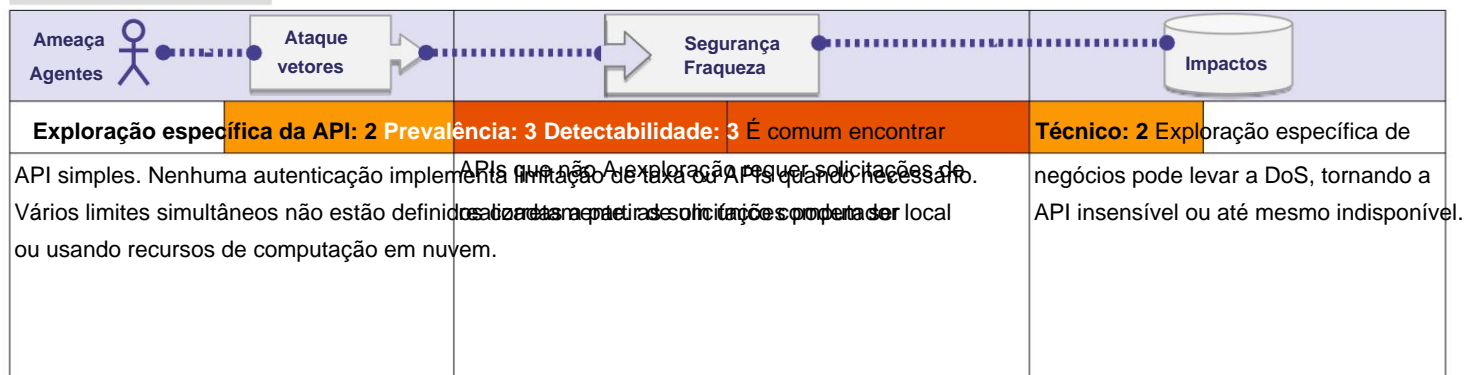
- Nunca confie no lado do cliente para filtrar dados confidenciais.
- Revise as respostas da API para garantir que contenham apenas dados legítimos.
- Os engenheiros de back-end devem sempre se perguntar "quem é o consumidor dos dados?" antes de expor um novo ponto de extremidade da API.
- Evite usar métodos genéricos como `to_json()` e `to_string()`. Em vez disso, escolha a dedo propriedades que você realmente deseja retornar.
- Classifique as informações confidenciais e de identificação pessoal (PII) que seu aplicativo armazena e com as quais trabalha, revisando todas as chamadas de API que retornam essas informações para ver se essas respostas representam um problema de segurança.
- Implemente um mecanismo de validação de resposta baseado em esquema como uma camada extra de segurança. Como parte disso O mecanismo define e aplica os dados retornados por todos os métodos da API, incluindo erros.

Referências

Externo

- [CWE-213: Exposição intencional de informações](#)

API4:2019 Falta de recursos e limitação de taxa



A API é vulnerável?

As solicitações de API consomem recursos como rede, CPU, memória e armazenamento. A quantidade de recursos necessária para atender a uma solicitação depende muito da entrada do usuário e da lógica de negócios do terminal. Além disso, considere o fato de que solicitações de vários clientes de API competem por recursos. Uma API é vulnerável se pelo menos um dos seguintes limites estiver ausente ou definido de forma inadequada (por exemplo, muito baixo/alto):

- Limites de tempo de execução
- Memória máxima alocável
- Número de descritores de arquivo
- Número de processos
- Tamanho da carga útil da solicitação (por exemplo, uploads)
- Número de solicitações por cliente/recurso
- Número de registros por página a serem retornados em uma única resposta de solicitação

Exemplos de Cenários de Ataque

Cenário 1

Um invasor carrega uma imagem grande emitindo uma solicitação POST para `/api/v1/images`. Quando o upload é concluído, a API cria várias miniaturas com tamanhos diferentes. Devido ao tamanho da imagem carregada, a memória disponível é esgotada durante a criação das miniaturas e a API deixa de responder.

Cenário #2

Temos uma aplicação que contém a lista de usuários em uma UI com limite de 200 usuários por página. A lista de usuários é recuperada do servidor usando a seguinte consulta: `/api/users?page=1&size=200`. Um invasor altera o parâmetro de tamanho para 200.000, causando problemas de desempenho no banco de dados. Enquanto isso, a API deixa de responder e não consegue lidar com outras solicitações deste ou de qualquer outro cliente (também conhecido como DoS).

O mesmo cenário pode ser usado para provocar erros de Estouro de Inteiro ou Estouro de Buffer.

API4:2019 Falta de recursos e limitação de taxa

Como prevenir

- O Docker facilita limitar a [memória, CPU, número de reinícios, descritores de arquivos e processos](#). • Implemente um limite na frequência com que um cliente pode chamar a API dentro de um período de tempo definido. • Notificar o cliente quando o limite for excedido, fornecendo o número limite e o horário em que o limite será redefinido.
- Adicione a validação adequada do lado do servidor para a string de consulta e os parâmetros do corpo da solicitação, especificamente aquele que controla o número de registros a serem retornados na resposta.
- Definir e aplicar o tamanho máximo de dados em todos os parâmetros de entrada e cargas úteis, como máximo comprimento para strings e número máximo de elementos em arrays.

Referências

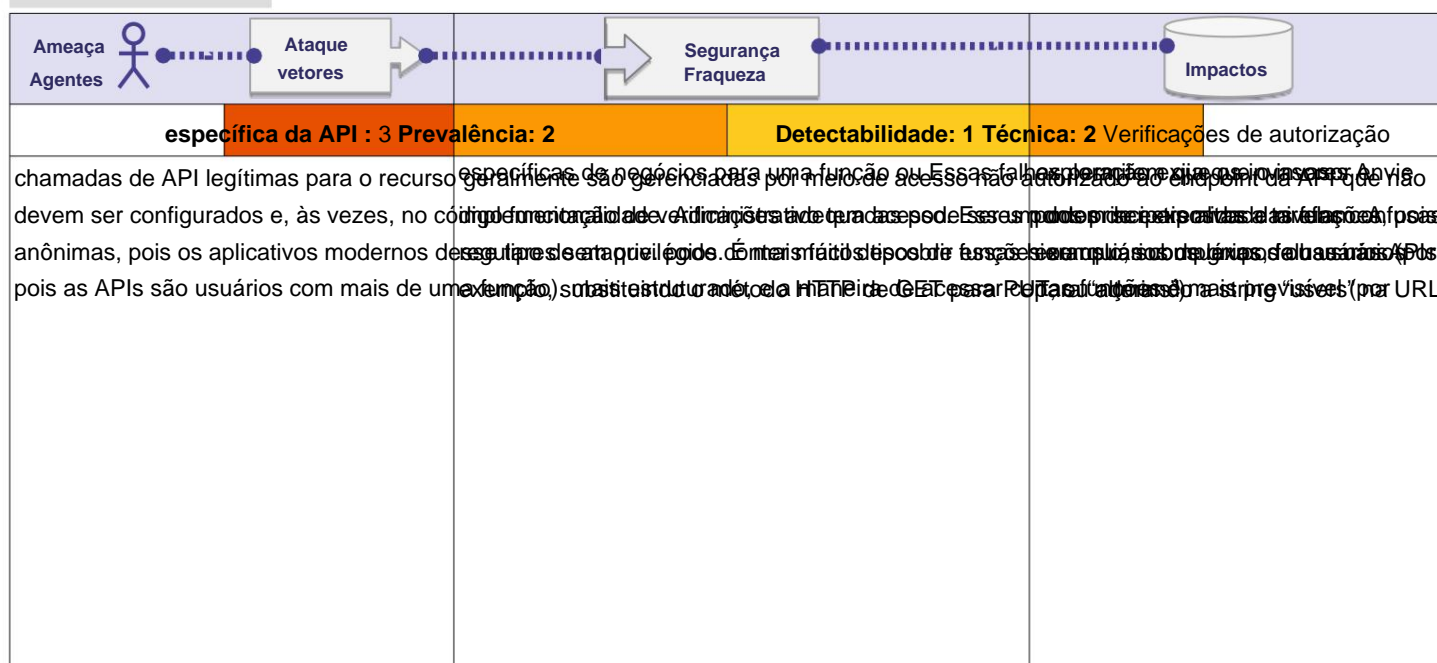
OWASP

- [Bloqueio de ataques de força bruta](#) • [Docker Cheat Sheet - Limitar recursos \(memória, CPU, descritores de arquivo, processos, reinicializações\)](#) • [Folha de Consulta de Avaliação REST](#)

Externo

- [CWE-307: Restrição imprópria de tentativas de autenticação excessivas](#) • [CWE-770: Alocação de Recursos Sem Limites ou Estrangulamento](#) • [“Rate Limiting \(Throttling\)” - Estratégias de segurança para sistemas de aplicativos baseados em microsserviços](#), NIST

API5:2019 Autorização de nível de função quebrada



A API é vulnerável?

A melhor maneira de encontrar problemas de autorização de nível de função quebrada é realizar uma análise profunda do mecanismo de autorização, tendo em mente a hierarquia do usuário, diferentes funções ou grupos no aplicativo e fazendo as seguintes perguntas:

- Um usuário comum pode acessar endpoints administrativos? •
- Um usuário pode executar ações confidenciais (por exemplo, criação, modificação ou exclusão) que não deveriam acesso simplesmente alterando o método HTTP (por exemplo, de GET para DELETE)? • Um
- usuário do grupo X pode acessar uma função que deveria ser exposta apenas para usuários do grupo Y, simplesmente adivinhando o URL e os parâmetros do ponto de extremidade (por exemplo, /api/v1/users/export_all)?

Não presuma que um endpoint de API seja regular ou administrativo apenas com base no caminho da URL.

Embora os desenvolvedores possam optar por expor a maioria dos pontos de extremidade administrativos em um caminho relativo específico, como api/admins, é muito comum encontrar esses pontos de extremidade administrativos em outros caminhos relativos junto com pontos de extremidade regulares, como api/users.

Exemplos de Cenários de Ataque

Cenário 1

Durante o processo de registro em um aplicativo que permite apenas a entrada de usuários convidados, o aplicativo móvel aciona uma chamada de API para GET /api/invites/{invite_guid}. A resposta contém um JSON com detalhes sobre o convite, incluindo a função e o e-mail do usuário.

Um invasor duplicou a solicitação e manipulou o método HTTP e o endpoint para POST /api/invites/new. Este endpoint deve ser acessado apenas por administradores usando o console administrativo, que não implementa verificações de autorização de nível de função.

O invasor explora o problema e envia a si mesmo um convite para criar uma conta de administrador:

```
POST /api/invites/new
{"email":"hugo@malicious.com","role":"admin"}
```

API5:2019 Autorização de nível de função quebrada

Cenário #2

Uma API contém um endpoint que deve ser exposto apenas para administradores - GET /api/admin/v1/users/all. Este terminal retorna os detalhes de todos os usuários do aplicativo e não implementa verificações de autorização em nível de função. Um invasor que aprendeu a estrutura da API dá um palpite e consegue acessar esse endpoint, que expõe detalhes confidenciais dos usuários do aplicativo.

Como prevenir

Seu aplicativo deve ter um módulo de autorização consistente e fácil de analisar que é chamado de todas as suas funções de negócios. Frequentemente, essa proteção é fornecida por um ou mais componentes externos ao código do aplicativo.

- O(s) mecanismo(s) de execução deve(m) negar todo o acesso por padrão, exigindo concessões explícitas para funções para acesso a todas as funções.
- Revise seus endpoints de API em relação a falhas de autorização de nível de função, tendo em mente a lógica de negócios da hierarquia de aplicativos e grupos.
- Certifique-se de que todos os seus controladores administrativos herdem de um controlador abstrato administrativo que implementa verificações de autorização com base no grupo/função do usuário.
- Certifique-se de que as funções administrativas dentro de um controlador regular implementem verificações de autorização com base no grupo e na função do usuário.

Referências

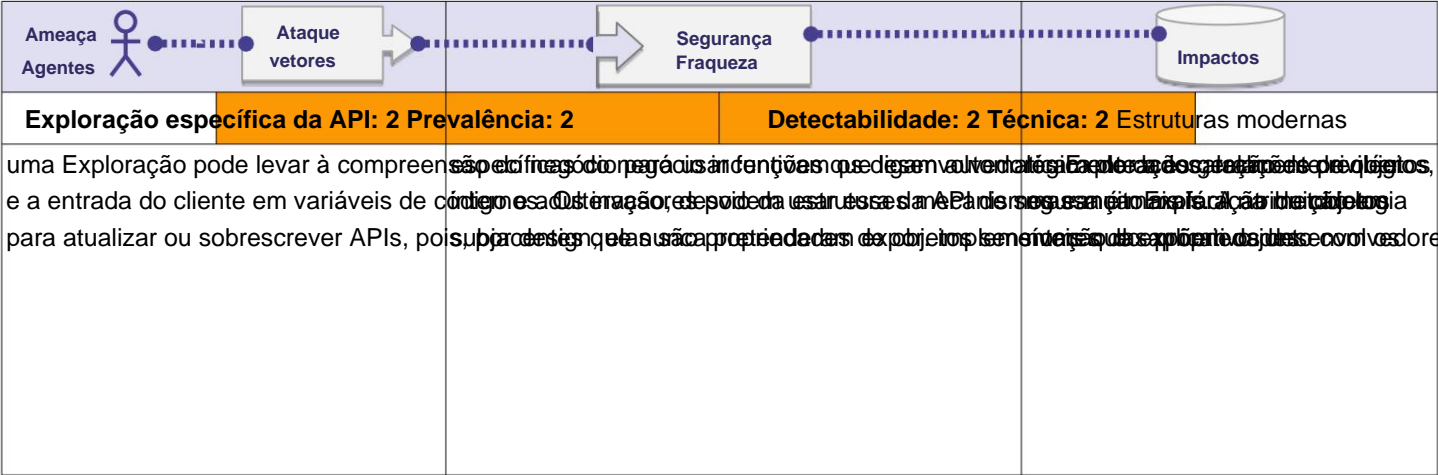
OWASP

- [Artigo OWASP sobre Navegação Forçada](#)
- [OWASP Top 10 2013-A7-Missing Function Level Access Control](#)
- [Guia de Desenvolvimento OWASP: Capítulo sobre Autorização](#)

Externo

- [CWE-285: Autorização imprópria](#)

API6:2019 Atribuição em massa



A API é vulnerável?

Objetos em aplicativos modernos podem conter muitas propriedades. Algumas dessas propriedades devem ser atualizadas diretamente pelo cliente (por exemplo, `user.first_name` ou `user.address`) e outras não (por exemplo, `user.is_vip` flag).

Um terminal de API é vulnerável se converter automaticamente os parâmetros do cliente em propriedades internas do objeto, sem considerar a sensibilidade e o nível de exposição dessas propriedades. Isso pode permitir que um invasor atualize as propriedades do objeto às quais ele não deveria ter acesso.

Exemplos de propriedades sensíveis:

- **Propriedades relacionadas à permissão:** `user.is_admin`, `user.is_vip` devem ser definidas apenas por administradores.
- **Propriedades dependentes do processo:** `user.cash` só deve ser definido internamente após a verificação do pagamento.
- **Propriedades internas:** `article.created_time` deve ser definido apenas internamente pelo aplicativo.

Exemplos de Cenários de Ataque

Cenário 1

Um aplicativo de compartilhamento de carona fornece ao usuário a opção de editar informações básicas para seu perfil. Durante esse processo, uma chamada de API é enviada para `PUT /api/v1/users/me` com o seguinte objeto JSON legítimo:

```
{"user_name": "inons", "age": 24}
```

A solicitação `GET /api/v1/users/me` inclui uma propriedade `credit_balance` adicional:

```
{"user_name": "inons", "age": 24, "credit_balance": 10}
```

O invasor repete a primeira solicitação com a seguinte carga útil:

```
{"user_name": "attacker", "age": 60, "credit_balance": 99999}
```

Como o endpoint é vulnerável à atribuição em massa, o invasor recebe créditos sem pagar.

API6:2019 Atribuição em massa

Cenário #2

Um portal de compartilhamento de vídeo permite que os usuários façam upload e download de conteúdo em diferentes formatos. Um invasor que explorou a API descobriu que o endpoint GET /api/v1/videos/{video_id}/meta_data retorna um objeto JSON com as propriedades do vídeo. Uma das propriedades é "mp4_conversion_params": "-v codec h264", que indica que o aplicativo usa um comando shell para converter o vídeo.

O invasor também descobriu que o terminal POST /api/v1/videos/new é vulnerável à atribuição em massa e permite que o cliente defina qualquer propriedade do objeto de vídeo. O invasor define um valor malicioso da seguinte forma: "mp4_conversion_params": "-v codec h264 && formato C:/"'. Esse valor causará uma injeção de comando shell assim que o invasor baixar o vídeo como MP4.

Como prevenir

- Se possível, evite usar funções que vinculam automaticamente a entrada de um cliente em variáveis de código ou objetos.
- Lista de permissões apenas as propriedades que devem ser atualizadas pelo cliente.
- Use recursos integrados para listar propriedades que não devem ser acessadas por clientes.
- Se aplicável, defina explicitamente e imponha esquemas para as cargas de dados de entrada.

Referências

Externo

- [CWE-915: Modificação indevidamente controlada de atributos de objetos determinados dinamicamente](#)

API7:2019 Configuração incorreta de segurança

Exploração	específica da API : 3	Prevalência: 3	Os invasores	Detectabilidade: 3	Técnico: 2	Específico para o negócio A
geralmente tentam encontrar falhas não corrigidas e extremidade comuns ou arquivos e diretórios de sistema legados para obter acesso não autorizado ou conhecimento do sistema.		configuração incorreta de segurança pode ocorrer em qualquer nível da pilha de aplicativos. Dados automatizados, mas também é possível que um invasor possa descobrir configurações totalmente incorretas do sistema.		uma API pode ser vulnerável se não for possível detectar e corrigir falhas de segurança. Isso pode ocorrer em qualquer nível da pilha de aplicativos, mas também é possível que um invasor possa descobrir configurações totalmente incorretas do sistema.		

A API é vulnerável?

A API pode estar vulnerável se:

- O fortalecimento de segurança apropriado está ausente em qualquer parte da pilha de aplicativos ou se foi permissões configuradas em serviços de nuvem.
- Faltam os patches de segurança mais recentes ou os sistemas estão desatualizados.
- Recursos desnecessários estão ativados (por exemplo, verbos HTTP).
- A segurança da camada de transporte (TLS) está ausente.
- As diretivas de segurança não são enviadas aos clientes (por exemplo, [cabeçalhos de segurança](#)).
- Uma política de compartilhamento de recursos entre origens (CORS) está ausente ou definida incorretamente.
- As mensagens de erro incluem rastreamentos de pilha ou outras informações confidenciais são expostas.

Exemplos de Cenários de Ataque

Cenário 1

Um invasor encontra o arquivo `.bash_history` no diretório raiz do servidor, que contém comandos usados pela equipe DevOps para acessar a API:

```
$ curl -X GET 'https://api.server/endpoint/' -H 'autorização: Basic Zm9vOmJhcG=='
```

Um invasor também pode encontrar novos endpoints na API que são usados apenas pela equipe de DevOps e não estão documentados.

Cenário #2

Para atingir um serviço específico, um invasor usa um mecanismo de pesquisa popular para procurar computadores acessíveis diretamente pela Internet. O invasor encontrou um host executando um sistema de gerenciamento de banco de dados popular, escutando na porta padrão. O host estava usando a configuração padrão, que tem a autenticação desativada por padrão, e o invasor obteve acesso a milhões de registros com PII, preferências pessoais e dados de autenticação.

Cenário #3

Ao inspecionar o tráfego de um aplicativo móvel, um invasor descobre que nem todo o tráfego HTTP é executado em um protocolo seguro (por exemplo, TLS). O invasor descobre que isso é verdade, especificamente para o download de imagens de perfil. Como a interação do usuário é binária, apesar do tráfego da API ser executado em um protocolo seguro, o invasor encontra um padrão no tamanho das respostas da API, que ele usa para rastrear as preferências do usuário sobre o conteúdo renderizado (por exemplo, imagens de perfil).

API7:2019 Configuração incorreta de segurança

Como prevenir

O ciclo de vida da API deve incluir:

- Um processo de endurecimento repetível que leva à implantação rápida e fácil de um bloqueio adequado meio Ambiente.
- Uma tarefa para revisar e atualizar as configurações em toda a pilha de API. A revisão deve incluir: arquivos de orquestração, componentes de API e serviços de nuvem (por exemplo, permissões de bucket do S3).
- Um canal de comunicação seguro para todas as interações de API, acesso a ativos estáticos (por exemplo, imagens).
- Um processo automatizado para avaliar continuamente a eficácia da configuração e configurações em todos ambientes.

Além disso:

- Para evitar que rastreamentos de exceção e outras informações valiosas sejam enviadas de volta aos invasores, se aplicável, defina e imponha todos os esquemas de carga útil de resposta da API, incluindo respostas de erro.
- Certifique-se de que a API só pode ser acessada pelos verbos HTTP especificados. Todos os outros verbos HTTP devem ser desativados (por exemplo, CABEÇA).
- As APIs que esperam ser acessadas de clientes baseados em navegador (por exemplo, WebApp front-end) devem implementar uma política adequada de compartilhamento de recursos entre origens (CORS).

Referências

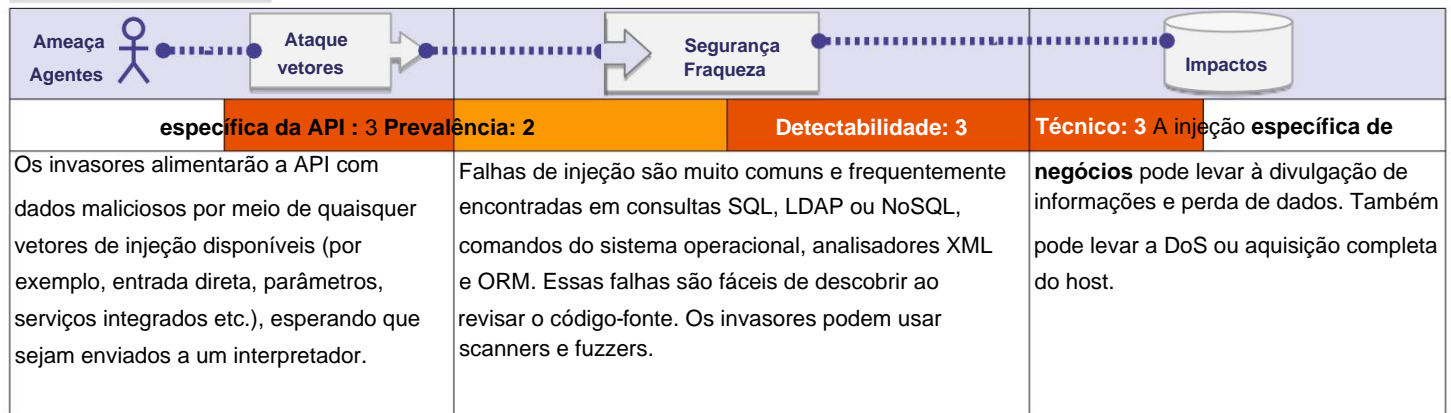
OWASP

- [Projeto de cabeçalhos seguros OWASP](#)
- [Guia de Teste OWASP: Gerenciamento de Configuração](#)
- [Guia de teste OWASP: teste de códigos de erro](#)
- [Guia de Teste OWASP: Teste o Compartilhamento de Recursos de Origem Cruzada](#)

Externo

- [CWE-2: Falhas de Segurança Ambiental](#)
- [CWE-16: Configuração](#)
- [CWE-388: Tratamento de Erros](#)
- [Guia para Segurança Geral do Servidor, NIST](#)
- [Let's Encrypt: uma autoridade de certificação gratuita, automatizada e aberta](#)

API8:2019 Injeção



A API é vulnerável?

A API é vulnerável a falhas de injeção se:

- Os dados fornecidos pelo cliente não são validados, filtrados ou sanitizados pela API.
- Os dados fornecidos pelo cliente são usados diretamente ou concatenados para consultas SQL/NoSQL/LDAP, comandos do sistema operacional, analisadores XML e Mapeamento Relacional de Objeto (ORM)/Mapeador de Documento de Objeto (ODM).
- Os dados provenientes de sistemas externos (por exemplo, sistemas integrados) não são validados, filtrados ou sanitizados por a API.

Exemplos de Cenários de Ataque

Cenário 1

O firmware de um dispositivo de controle parental fornece o endpoint `/api/CONFIG/restore` que espera que um `appid` seja enviado como um parâmetro multipartes. Usando um descompilador, um invasor descobre que o `appid` é passado diretamente para uma chamada do sistema sem qualquer sanitização:

```
snprintf(cmd, 128, "%srestore_backup.sh /tmp/postfile.bin %s %d",
        "/mnt/shares/usr/bin/scripts/", appid, 66); sistema(cmd);
```

O seguinte comando permite que o invasor desligue qualquer dispositivo com o mesmo firmware vulnerável:

```
$ curl -k "https://{deviceIP}:4567/api/CONFIG/restore" -F 'appid=$(/etc/pod/
power_down.sh)'
```

Cenário #2

Temos um aplicativo com funcionalidade básica de CRUD para operações com reservas. Um invasor conseguiu identificar que a injeção de NoSQL pode ser possível por meio do parâmetro de string de consulta `bookingId` na solicitação de exclusão de agendamento. A solicitação fica assim: `DELETE /api/bookings?bookingId=678`.

O servidor da API usa a seguinte função para lidar com solicitações de exclusão:

```
router.delete('/bookings', função assíncrona (req, res, próximo) {
  tente
    { const deleteBooking = await Bookings.findOneAndRemove({_id : req.query.bookingId}); res.status(200); } catch (err)
    { res.status(400).json({ error: 'Ocorreu um erro inesperado ao processar uma solicitação' });
  }
});
```

API8:2019 Injeção

O invasor interceptou a solicitação e alterou o parâmetro de string de consulta bookingId conforme mostrado abaixo. Nesse caso, o invasor conseguiu excluir a reserva de outro usuário:

EXCLUIR /api/bookings?bookingId[\$ne]=678

Como prevenir

Prevenir a injeção requer manter os dados separados dos comandos e consultas.

- Execute a validação de dados usando uma biblioteca única, confiável e mantida ativamente.
- Valide, filtre e limpe todos os dados fornecidos pelo cliente ou outros dados provenientes de sistemas integrados.
- Caracteres especiais devem ser escapados usando a sintaxe específica para o interpretador de destino.
- Prefira uma API segura que forneça uma interface parametrizada.
- Sempre limite o número de registros devolvidos para evitar divulgação em massa em caso de injeção.
- Valide os dados de entrada usando filtros suficientes para permitir apenas valores válidos para cada parâmetro de entrada.
- Definir tipos de dados e padrões rígidos para todos os parâmetros de string.

Referências

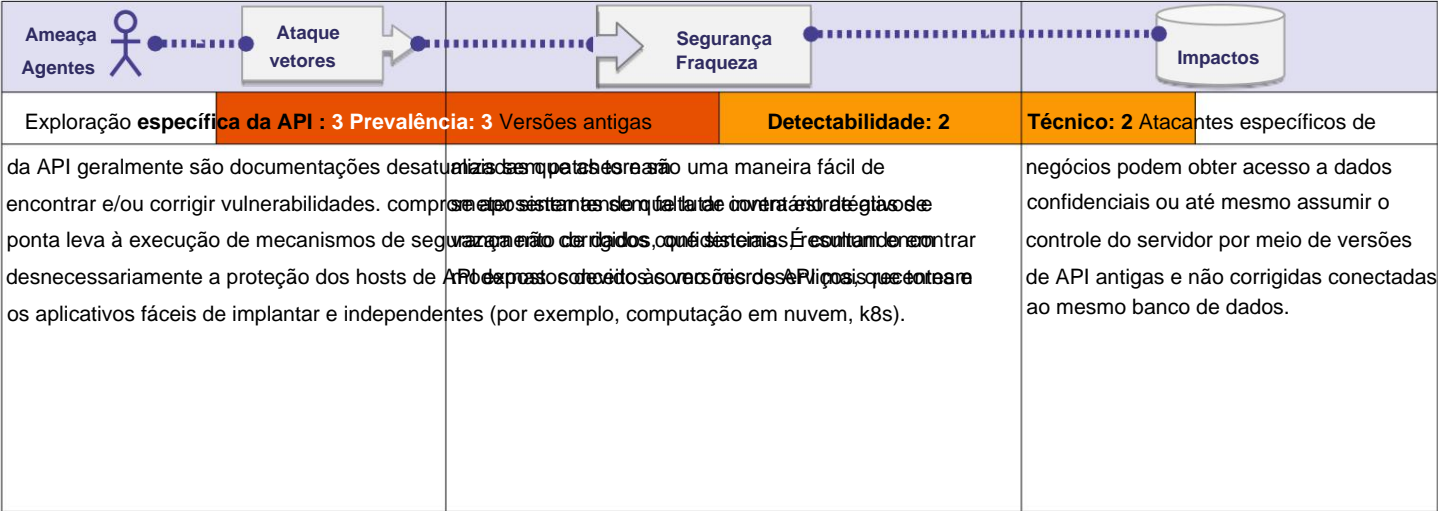
OWASP

- [Falhas de Injeção OWASP](#)
- [Injeção de SQL](#)
- [Diversão com injeção NoSQL com objetos e matrizes](#)
- [Injeção de comando](#)

Externo

- [CWE-77: Injeção de Comando](#)
- [CWE-89: Injeção de SQL](#)

API9:2019 Gerenciamento de ativos impróprios



A API é vulnerável?

A API pode estar vulnerável se:

- O objetivo de um host de API não é claro e não há respostas explícitas para as seguintes perguntas: • Em qual ambiente a API está sendo executada (por exemplo, produção, preparação, teste, desenvolvimento)? • Quem deve ter acesso de rede à API (por exemplo, público, interno, parceiros)? • Qual versão da API está em execução? • Quais dados são coletados e processados pela API (por exemplo, PII)? • Qual é o fluxo de dados?
- Não há documentação ou a documentação existente não está atualizada. • Não há plano de aposentadoria para cada versão da API. • O inventário de hosts está ausente ou desatualizado. • Inventário de serviços integrados, próprios ou de terceiros, ausente ou desatualizado. • Versões de API antigas ou anteriores estão sendo executadas sem correção.

Exemplos de Cenários de Ataque

Cenário 1

Depois de redesenhar seus aplicativos, um serviço de pesquisa local deixou uma versão antiga da API (api.someservice.com/v1) em execução, desprotegida e com acesso ao banco de dados do usuário. Ao visar um dos aplicativos lançados mais recentemente, um invasor encontrou o endereço da API (api.someservice.com/v2).

A substituição de v2 por v1 na URL deu ao invasor acesso à API antiga e desprotegida, expondo as informações de identificação pessoal (PII) de mais de 100 milhões de usuários.

Cenário #2

Uma rede social implementou um mecanismo de limitação de taxa que impede que invasores usem força bruta para adivinhar tokens de senha de redefinição. Este mecanismo não foi implementado como parte do próprio código da API, mas em um componente separado entre o cliente e a API oficial (www.socialnetwork.com). Um pesquisador encontrou um host de API beta (www.mbasic.beta.socialnetwork.com) que executa a mesma API, incluindo o mecanismo de redefinição de senha, mas o mecanismo de limitação de taxa não estava em vigor. O pesquisador conseguiu redefinir a senha de qualquer usuário usando uma simples força bruta para adivinhar o token de 6 dígitos.

API9:2019 Gerenciamento de ativos impróprios

Como prevenir

- Inventariar todos os hosts de API e documentar aspectos importantes de cada um deles, com foco na API ambiente (por exemplo, produção, preparação, teste, desenvolvimento), quem deve ter acesso de rede ao host (por exemplo, público, interno, parceiros) e a versão da API.
- Inventariar serviços integrados e documentar aspectos importantes, como sua função no sistema, quais dados é trocado (fluxo de dados) e sua sensibilidade. •

Documente todos os aspectos de sua API, como autenticação, erros, redirecionamentos, limitação de taxa, política de compartilhamento de recursos de origem cruzada (CORS) e endpoints, incluindo seus parâmetros, solicitações e respostas. • Gerar documentação automaticamente adotando padrões abertos. Inclua a compilação da documentação em seu pipeline de CI/CD.

- Disponibilizar a documentação da API para aqueles autorizados a usar a API. •

Use medidas de proteção externa, como firewalls de segurança de API para todas as versões expostas de suas APIs, não apenas para a versão de produção atual. •

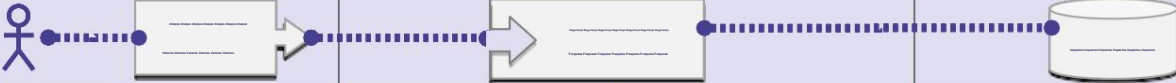
Evite usar dados de produção com implantações de API que não sejam de produção. Se isso for inevitável, esses endpoints devem receber o mesmo tratamento de segurança que os de produção. • Quando versões mais recentes de APIs incluírem melhorias de segurança, realize análises de risco para fazer a decisão das ações de mitigação necessárias para a versão mais antiga: por exemplo, se é possível fazer o backport das melhorias sem quebrar a compatibilidade da API ou se você precisa retirar a versão mais antiga rapidamente e forçar todos os clientes a migrar para a versão mais recente.

Referências

Externo

- [CWE-1059: Documentação incompleta](#)
- [Iniciativa OpenAPI](#)

API10:2019 Registro e monitoramento insuficientes

				
Exploração específica da API: 2	Prevalência: 3	Os invasores	Detectabilidade: 1	Técnico: 2
aproveitam a falta de registro e monitoramento em sistemas sem serem notados.		Sem registro e monitoramento, ou com registro e monitoramento insuficientes, é quase impossível rastrear atividades suspeitas e responder a elas em tempo hábil.		Sem visibilidade das atividades maliciosas em andamento, os invasores têm muito tempo para comprometer totalmente os sistemas.

A API é vulnerável?

A API é vulnerável se:

- Não produz nenhum log, o nível de log não está definido corretamente ou as mensagens de log não incluem detalhes suficientes.
- A integridade do log não é garantida (por exemplo, [injeção de log](#)).
- Os logs não são monitorados continuamente.
- A infraestrutura da API não é monitorada continuamente.

Exemplos de Cenários de Ataque

Cenário 1

As chaves de acesso de uma API administrativa vazaram em um repositório público. O proprietário do repositório foi notificado por e-mail sobre o possível vazamento, mas levou mais de 48 horas para agir sobre o incidente, e a exposição das chaves de acesso pode ter permitido o acesso a dados confidenciais. Devido ao registro insuficiente, a empresa não consegue avaliar quais dados foram acessados por agentes mal-intencionados.

Cenário #2

Uma plataforma de compartilhamento de vídeo foi atingida por um ataque de preenchimento de credenciais de “grande escala”. Apesar de logins com falha serem registrados, nenhum alerta foi acionado durante o intervalo de tempo do ataque. Como reação às reclamações dos usuários, os logs da API foram analisados e o ataque foi detectado. A empresa teve que fazer um anúncio público pedindo aos usuários que redefinam suas senhas e relatar o incidente às autoridades reguladoras.

Como prevenir

- Registre todas as tentativas de autenticação com falha, acesso negado e erros de validação de entrada.
- Os logs devem ser gravados usando um formato adequado para serem consumidos por uma solução de gerenciamento de log e devem incluir detalhes suficientes para identificar o ator mal-intencionado.
- Os logs devem ser tratados como dados confidenciais e sua integridade deve ser garantida em repouso e em trânsito.
- Configurar um sistema de monitoramento para monitorar continuamente a infraestrutura, a rede e a API funcionando.
- Use um sistema de gerenciamento de eventos e informações de segurança (SIEM) para agregar e gerenciar logs de todos os componentes da pilha e hosts da API.
- Configurar painéis e alertas personalizados, permitindo que atividades suspeitas sejam detectadas e respondidas mais cedo.

Referências

OWASP

- [Folha de dicas de registro OWASP](#)
- [Controles proativos OWASP: implementação de registro e detecção de intrusão](#)
- [Padrão de verificação de segurança de aplicativos OWASP: V7: tratamento de erros e verificação de registro](#)
[Requisitos](#)

Externo

- [CWE-223: Omissão de informações relevantes para a segurança](#)
- [CWE-778: registro insuficiente](#)

+D O que vem a seguir para desenvolvedores

A tarefa de criar e manter um software seguro ou consertar um software existente pode ser difícil. As APIs não são diferentes.

Acreditamos que a educação e a conscientização são fatores-chave para escrever um software seguro. Tudo o mais necessário para atingir a meta depende do **estabelecimento e uso de processos de segurança repetíveis e controles de segurança padrão**.

A OWASP possui inúmeros recursos gratuitos e abertos para abordar a segurança desde o início do projeto.

Visite a [página de Projetos OWASP](#) para obter uma lista abrangente de projetos disponíveis.

Educação	Você pode começar a ler os materiais do Projeto de Educação OWASP de acordo com sua profissão e interesse. Para aprendizado prático, adicionamos crAPI - API Completamente Ridícula em nosso roteiro . Enquanto isso, você pode praticar WebAppSec usando o módulo OWASP DevSlop Pixi , um WebApp vulnerável e serviço de API com a intenção de ensinar aos usuários como testar aplicativos da Web modernos e APIs para problemas de segurança e como escrever APIs mais seguras no futuro. Você também pode participar da Conferência OWASP AppSec sessões de treinamento ou junte-se ao seu capítulo local .
Requisitos de segurança	A segurança deve fazer parte de todos os projetos desde o início. Ao fazer a elicitação de requisitos, é importante definir o que significa "seguro" para aquele projeto. OWASP recomenda que você use o OWASP Application Security Verification Standard (ASVS) como um guia para definir os requisitos de segurança . Se você estiver terceirizando , considere o Anexo do contrato de software seguro OWASP , que devem ser adaptados de acordo com as leis e regulamentos locais.
Arquitetura de segurança	A segurança deve continuar sendo uma preocupação durante todas as etapas do projeto. As folhas de dicas de prevenção OWASP são um bom ponto de partida para orientação sobre como projetar a segurança durante a fase de arquitetura. Entre muitos outros, você encontrará a folha de dicas de segurança REST e a folha de dicas de avaliação REST .
Segurança Padrão Controles	A adoção de Controles de segurança padrão reduz o risco de apresentar falhas de segurança ao escrever sua própria lógica. Apesar do fato de que muitas estruturas modernas agora vêm com controles efetivos padrão integrados, o OWASP Proactive Controls fornece uma boa visão geral de quais controles de segurança você deve procurar incluir em seu projeto. OWASP também fornece algumas bibliotecas e ferramentas que você pode achar valiosas, como controles de validação.
Software seguro Ciclo de vida do desenvolvimento	Você pode usar o Software Assurance Maturity Model (SAMM) da OWASP para melhorar o processo ao criar APIs. Vários outros projetos OWASP estão disponíveis para ajudá-lo durante as diferentes fases de desenvolvimento da API, por exemplo, o OWASP Code Review Project .

+DSO O que vem a seguir para DevSecOps

Devido à sua importância nas arquiteturas de aplicativos modernos, a construção de APIs seguras é crucial. A segurança não pode ser negligenciada e deve fazer parte de todo o ciclo de vida do desenvolvimento. Varreduras e testes de penetração anuais não são mais suficientes.

O DevSecOps deve se juntar ao esforço de desenvolvimento, facilitando o teste de segurança contínuo em todo o ciclo de vida do desenvolvimento de software. Seu objetivo é aprimorar o pipeline de desenvolvimento com automação de segurança e sem afetar a velocidade do desenvolvimento.

Em caso de dúvida, mantenha-se informado e consulte o [DevSecOps Manifesto](#) frequentemente.

Entenda a ameaça Modelo	As prioridades de teste vêm de um modelo de ameaça. Se você não tiver um, considere usar o OWASP Application Security Verification Standard (ASVS) , e o Guia de Teste OWASP como uma entrada. Envolver a equipe de desenvolvimento pode ajudar a torná-los mais conscientes da segurança.
Entenda o SDLC	Junte-se à equipe de desenvolvimento para entender melhor o ciclo de vida do desenvolvimento de software. Sua contribuição em testes contínuos de segurança deve ser compatível com pessoas, processos e ferramentas. Todos devem concordar com o processo, para que não haja atrito ou resistência desnecessários.
Estratégias de teste	Como seu trabalho não deve afetar a velocidade de desenvolvimento, você deve escolher com sabedoria a melhor técnica (simples, rápida e precisa) para verificar os requisitos de segurança. A estrutura de conhecimento de segurança OWASP e padrão de verificação de segurança de aplicativos OWASP podem ser ótimas fontes de requisitos de segurança funcionais e não funcionais. Existem outras ótimas fontes para projetos e ferramentas semelhante ao oferecido pela comunidade DevSecOps .
Obtenção de Cobertura e Precisão	Você é a ponte entre os desenvolvedores e as equipes de operações. Para obter cobertura, você deve se concentrar não apenas na funcionalidade, mas também na orquestração. Trabalhe próximo às equipes de desenvolvimento e operações desde o início para otimizar seu tempo e esforço. Você deve buscar um estado em que a segurança essencial seja verificada continuamente.
Comunique-se com clareza Descobertas	Contribua com valor com menos ou nenhum atrito. Entregue as descobertas em tempo hábil, dentro das ferramentas que as equipes de desenvolvimento estão usando (não em arquivos PDF). Junte-se à equipe de desenvolvimento para abordar as descobertas. Aproveite a oportunidade para educá-los, descrevendo claramente a fraqueza e como ela pode ser abusada, incluindo um cenário de ataque para torná-la real.

+DAT Metodologia e Dados

Visão geral

Como a indústria de AppSec não tem sido focada especificamente na arquitetura mais recente de aplicativos, na qual as APIs desempenham um papel importante, compilar uma lista dos dez riscos de segurança de API mais críticos, com base em uma chamada pública de dados, teria sido uma tarefa difícil tarefa. Apesar de não haver chamada de dados públicos, a lista dos 10 principais resultante ainda é baseada em dados disponíveis publicamente, contribuições de especialistas em segurança e discussão aberta com a comunidade de segurança.

Metodologia e Dados

Na primeira fase, dados disponíveis publicamente sobre incidentes de segurança de APIs foram coletados, revisados e categorizados por um grupo de especialistas em segurança. Esses dados foram coletados de plataformas de recompensas de bugs e bancos de dados de vulnerabilidades, dentro de um período de um ano. Foi usado para fins estatísticos.

Na próxima fase, os profissionais de segurança com experiência em testes de penetração foram solicitados a compilar sua própria lista dos 10 principais.

A [Metodologia de Classificação de Risco OWASP](#) foi usado para realizar a Análise de Risco. As pontuações foram discutidas e revisadas entre os profissionais de segurança. Para considerações sobre esses assuntos, consulte a seção [API Security Risks](#).

O primeiro rascunho do OWASP API Security Top 10 2019 resultou de um consenso entre os resultados estatísticos da primeira fase e as listas dos profissionais de segurança. Este rascunho foi então submetido à apreciação e revisão por outro grupo de profissionais de segurança, com experiência relevante nas áreas de segurança de APIs.

O OWASP API Security Top 10 2019 foi apresentado pela primeira vez no evento OWASP Global AppSec Tel Aviv (maio de 2019). Desde então, está disponível no GitHub para discussão e contribuições públicas.

A lista de colaboradores está disponível na seção [Agradecimentos](#).

+ACK Agradecimentos

Agradecimentos aos Colaboradores

Gostaríamos de agradecer aos seguintes colaboradores que contribuíram publicamente no GitHub ou por outros meios:

- 007divyachawla
- Abid Khan • Adam Fisher
- outroik
- bkimminich
- caseysoftware
- Chris Westphal
- dsopas • DSotnikov
- emilva
- ErezYalon
- flasceles
- Guillaume Benats
- IgorSasovets
- Inonshk • JonnySchnittger • jmanico • jmdx • Keith Casey • kozmic

- LauraRosePorter
- Matthieu Estrade
- nathanawmk
- Paulo ASilva
- pentagramz
- philippederyck • pleothaud • r00ter

- Raj Kumar
- Sagar Popat
- Stephen Gates
- thomaskonrad
- xycloops123