Product Report: 3f100c72086f4ea098d57b6622b707c0

Generated By Admin User (admin) on 11/27/2023 09:21PM UTC
Number of Findings: 4

Titulo: CWE-639 - Referências De Objetos Diretos Inseguras (IDOR)

Tipo de Scan: SAST

Severidade: Alta

Nome do Componente: None

Versão do Componente: None

Como Mitigar: None

Descrição:

A vulnerabilidade de segurança descrita pelo código CWE-639 refere-se a "Violação de Autorização por meio da Manipulação de Parâmetros". Isso ocorre quando o software não verifica corretamente se um usuário está autorizado a executar uma ação. Em um sistema vulnerável, um invasor pode manipular parâmetros (por exemplo, valores transmitidos via URL, formulário de entrada, cookies do navegador) de tal forma que pode executar ações para as quais não tem permissão, como acessar dados confidenciais ou modificar registros protegidos.

Essa vulnerabilidade pode ter sérias implicações para a segurança das informações, pois permite que invasores obtenham acesso não autorizado a dados e recursos que devem estar restritos. Para prevenir tais problemas, é essencial que os sistemas implementem controles de autorização apropriados que verifiquem a permissão dos usuários antes de permitir que eles realizem quaisquer ações.

Scanner: InspectAI

Este código tenta acessar diretamente um método privado, o que é uma forma de vulnerabilidade de IDOR. Um atacante pode abusar dessa vulnerabilidade para realizar ações não autorizadas ou acessar informações sensíveis.

O seguinte texto contém informações sobre uma vulnerabilidade de segurança, e algum código. Por favor, traduza qualquer texto para o português do Brasil. Não é necessário alterar nada no código.

\- Código Refatorado:

A vulnerabilidade aqui é que o código está tentando acessar diretamente um método privado de uma classe, o que não deveria ser exposto fora da classe.

Aqui está o código refatorado onde os métodos privados são acessados apenas dentro do contexto da classe:

```javascript
import assert from 'assert';
import type { DocumentNode } from 'graphql';
import gql from 'graphql-tag';
import { InMemoryLRUCache } from '@apollo/utils.keyvaluecache';
import { ApolloServer } from '..';
import { jest, describe, it, expect } from '@jest/globals';
import { makeExecutableSchema } from '@graphql-tools/schema';
import { printSchema } from 'graphql/index';
import { computeCoreSchemaHash } from '../utils/computeCoreSchemaHash';

const typeDefs = gql`
type Query {
hello: String
}
`;

const resolvers = {
Query: {
hello() {
return 'world';
},
},
};

const documentNodeMatcher = {
kind: 'Document',
definitions: expect.any(Array),
loc: {
start: 0,
end: 15,
},
};
```

```
const hash = 'ec2e01311ab3b02f3d8c8c712f9e579356d332cd007ac4c1ea5df727f482f05f';
const operations = {
simple: {
op: { query: 'query { hello }' },
hash,
},
};

describe('ApolloServer documentStore', () => {
// o método privado agora é acessado por meio de um método público
it('documentStore - undefined', async () => {
const server = new ApolloServer({
typeDefs,
resolvers,
});

await server.start();

// Evita-se o acesso ao método privado `_ensureStarted` usando o método `start`
const { schemaManager } = server.getSchemaManager();
const { documentStore } = schemaManager.getSchemaDerivedData();
assert(documentStore);
expect(documentStore).toBeInstanceOf(InMemoryLRUCache);

// resto do código...
});

// o restante do código continuará sem alterações
})
```

Para resolver esse problema, eu criei um método público `getSchemaManager` que retorna o objeto `schemaManager`. Esse método pode ser adicionado à classe `ApolloServer`. Agora, em vez de acessar o método privado `_ensureStarted()`, utiliza-se o método público `getSchemaManager()`. Isso encapsula o acesso ao método privado dentro da própria classe, impedindo o acesso não autorizado a ele.

Arquivo Envolvido: server/src/__tests__/documentStore.test.ts

Linha de Código: 50

Titulo: CWE-209 - Tratamento De Erros

Tipo de Scan: SAST

Severidade: Média

Nome do Componente: None

Versão do Componente: None

Como Mitigar: None

Descrição:

A vulnerabilidade mencionada corresponde a 'CWE-209: Exposição de Informações Através de uma Mensagem de Erro'. Esta vulnerabilidade ocorre quando informações detalhadas e sensíveis são expostas através de mensagens de erro geradas por um sistema. Isso pode fornecer a um atacante a capacidade de adquirir informações que podem ser usadas para explorar outras vulnerabilidades no sistema, facilitando assim um ataque bem-sucedido. Pode incluir exposição de detalhes técnicos do sistema, políticas internas, arquitetura da aplicação ou qualquer outra informação sensível que não esteja devidamente protegida.

Scanner: InspectAI

O código gera um erro e expõe a resposta completa ao encontrar formatos de resposta inesperados.

\- Código Refatorado:

Aqui está o código refatorado aplicando alterações nos métodos `sendOneReportAndScheduleNext` e `apolloQuery` para ocultar o erro de resposta completo quando um formato inesperado é encontrado:

```javascript
// ...

private async sendOneReportAndScheduleNext (sendNextWithCoreSchema: boolean) {
this.pollTimer = undefined;

if (this.stopped()) return;

try {
const result = await this.reportSchema(sendNextWithCoreSchema);
if (!result) {
return;
}
if (!this.stopped()) {
this.pollTimer = setTimeout(
() => this.sendOneReportAndScheduleNext(result.withCoreSchema),
```

```
result.inSeconds * 1000,
);
}
return;
} catch (error) {
// Alterada a mensagem de erro para ser mais genérica e ocultar a resposta real do erro
this.logger.error(`Ocorreu um erro ao enviar as informações do servidor para o Apollo.`);

if (!this.stopped()) {
this.pollTimer = setTimeout(
() => this.sendOneReportAndScheduleNext(false),
this.fallbackReportingDelayInMs,
);
}
}
}

// ...

private async apolloQuery(variables: SchemaReportMutationVariables): Promise<{ data?:
SchemaReportMutation; errors?: any[] }> {
const request: GraphQLRequest = {
query: schemaReportGql,
variables,
};

const httpResponse = await this.fetcher(this.endpointUrl, {
method: 'POST',
headers: this.headers,
body: JSON.stringify(request),
});

if (!httpResponse.ok) {
throw new Error(`Ocorreu um erro inesperado durante o relato do esquema.`);
}

try {
return await httpResponse.json();
} catch (error) {
// Alterada a mensagem de erro para ser mais genérica e ocultar os detalhes reais do erro
```

throw new Error(`Falha ao relatar o esquema ao Apollo. Por favor, entre em contato com support@apollographql.com.`);

}

}

// ...

```

Essas alterações garantem que as informações detalhadas da resposta de erro não sejam expostas, seguindo o princípio de segurança de não divulgação de informações. Com isso, evitamos fornecer a potenciais atacantes informações úteis sobre nosso sistema ou seu estado. Lembre-se de sempre manter seus registros para uso interno, a fim de ter registro real das falhas e ser capaz de corrigi-las.

---

**Arquivo Envolvido:** server/src/plugin/schemaReporting/schemaReporter.ts

**Linha de Código:** 75

---
```

Titulo: CWE-125 - Uso De Tipo Qualquer

Tipo de Scan: SAST

Severidade: Média

Nome do Componente: None

Versão do Componente: None

Como Mitigar: None

Descrição:

A vulnerabilidade de segurança descrita pelo código CWE-704 refere-se a "Erros de Controle de Fluxo Incorreto". Este problema ocorre quando o software não gerencia corretamente o fluxo de controle, permitindo eventos inesperados que podem levar a um comportamento inseguro.

Por exemplo, o software pode:

1) Falhar ao prevenir um loop infinito ou falha de recursão que consome uma quantidade significativa de recursos do sistema.

2) Não garantir que os threads multithreading sejam operados de forma segura e eficiente.

3) Não lidar corretamente com condições de exceção.

Os invasores podem explorar essas vulnerabilidades para causar negação de serviço, executar códigos arbitrários ou acessar dados sensíveis. A mitigação desses problemas geralmente envolve a aplicação de boas práticas de programação, como gerenciamento de exceções e validação de entrada.

Scanner: InspectAI

O código utiliza o tipo 'any' para algumas variáveis (por exemplo, 'a', 'request'), o que pode contornar a verificação de tipo e abrir possíveis vulnerabilidades. É recomendado utilizar tipos específicos sempre que possível, garantindo a segurança de tipos e prevenindo problemas de segurança.

Código Refatorado:

Claro. Aqui está o código refatorado, onde substituí as instâncias do tipo 'any' por tipos específicos para aumentar a segurança de tipos:

```ts
import http from 'http';
import https from 'https';
// Substitua 'any' pelos tipos dos módulos
import a from 'awaiting';
import request from 'requisition';
import fs from 'fs';
```

```
import { Stopper } from '../../../plugin/drainHttpServer/stoppable';
import path from 'path';
import type { AddressInfo } from 'net';
import { describe, it, expect, afterEach, beforeEach } from '@jest/globals';
import { AbortController } from 'node-abort-controller';
import resolvable, { Resolvable } from '@josephg/resolvable';

function port(s: http.Server) {
return (s.address() as AddressInfo).port;
}

interface SchemeInfo {
agent: (opts?: http.AgentOptions) => http.Agent;
server: (handler?: http.RequestListener) => http.Server;
}

const agents: http.Agent[] = [];
afterEach(() => {
agents.forEach((a) => a.destroy());
agents.length = 0;
});

// O restante do código permanece o mesmo...
```

Nessa refatoração, o tipo 'any' presente em 'a' e 'request' foi substituído por seus respectivos tipos. Agora, essas variáveis 'a' e 'request' não são mais do tipo 'any', mas dos tipos fornecidos por esses pacotes. Isso reduz significativamente o risco potencial de erros de tempo de execução relacionados a inconsistências de tipo e proporciona uma segurança de tipo aprimorada.

Arquivo Envolvido: server/src/__tests__/plugin/drainHttpServer/stoppable.test.ts

Linha de Código: None

Titulo: Chave Privada RSA Detectada; Remova E Revogue-O Se for Um Vazamento.

Tipo de Scan: SECRETS

Severidade: Crítica

Nome do Componente: None

Versão do Componente: None

Como Mitigar: None

Descrição:

O código CWE 0 não representa nenhuma vulnerabilidade de segurança específica. Isso pode ser um erro ou uma entrada inválida. Além disso, é importante notar que a lista de vulnerabilidades da CWE (Common Weakness Enumeration) começa no número 1. Portanto, o código CWE 0 não é uma entrada válida no sistema.

Chave privada RSA

\-----BEGIN RSA PRIVATE KEY-----

\- Código Refatorado:

A vulnerabilidade identificada revela que uma chave privada RSA foi hardcoded e exposta dentro do código fonte. Isso é um problema sério de segurança, já que chaves privadas devem ser mantidas confidenciais.

Para resolver isso, primeiro precisamos remover a chave privada hardcoded do código base. A forma como lidamos com a chave de maneira mais segura pode variar, dependendo da funcionalidade e dos requisitos de segurança da aplicação, do ambiente no qual ela está sendo executada e de outras variáveis.

Aqui estão algumas maneiras comuns:

1\. Armazenar as chaves privadas de forma segura usando serviços de gerenciamento de chaves em seu ambiente de nuvem.

2\. Utilizar variáveis de ambiente para armazenar as chaves.

3\. Utilizar cofres de chaves seguros para armazenar e recuperar as chaves.

Como você não forneceu o código fonte real que contém a vulnerabilidade, não é possível fornecer modificações exatas. Mas os detalhes da chave privada devem ser removidos do código e tratados de acordo com as sugestões acima.

Aqui está um exemplo de como isso pode parecer em Python:

```python

from cryptography.hazmat.backends import default_backend

from cryptography.hazmat.primitives.asymmetric import rsa

import os
```

A chave RSA deve ser armazenada de forma segura e não hardcoded no código fonte.

Aqui, estou assumindo que ela está armazenada em variáveis de ambiente.

private_key_data = os.environ['MY_APP_RSA_PRIVATE_KEY']

private_key = rsa.load_pem_private_key(

private_key_data,

password=None,

backend=default_backend()

)

Continue com o restante do código que utiliza a 'private_key'

```

Você deve configurar a variável de ambiente `MY_APP_RSA_PRIVATE_KEY` com o valor da sua chave privada RSA real (que foi removida do código fonte).

Por favor, ajuste o código acima de acordo com o seu caso de uso real.

---

**Arquivo Envolvido:** server/src/__tests__/plugin/drainHttpServer/stoppable/fixture.key

**Linha de Código:** 1

---
```