Product Report: 6045d50bb19d48278f0791adaec7a169

Generated By Admin User (admin) on 12/02/2023 01:58AM UTC
Number of Findings: 3

Titulo: CWE-xxx - Fracassa Se a Função De Fábrica Não Retornar Um Esquema

Tipo de Scan: SAST

Severidade: Baixa

Nome do Componente: None

Versão do Componente: None

Como Mitigar: None

Descrição:

A CWE-135 se refere a uma vulnerabilidade de inicialização. Essa vulnerabilidade ocorre quando um programa ou sistema de computador inicia em um estado inseguro, permitindo que um atacante tenha acesso não autorizado ao sistema. Isso pode acontecer, por exemplo, quando os controles de acesso não são implementados corretamente durante o processo de inicialização. Os invasores podem se aproveitar desse momento vulnerável para executar comandos maliciosos, modificar ou obter acesso a dados sensíveis, ou até mesmo assumir o controle completo do sistema. É importante que os desenvolvedores e equipes de segurança identifiquem e corrijam essa vulnerabilidade durante o processo de desenvolvimento e implementação de sistemas, a fim de garantir a segurança adequada. Medidas como autenticação segura, criptografia e validações adequadas podem ajudar a mitigar essa vulnerabilidade.

Scanner: InspectAI

O código cria uma função de fábrica de esquema que intencionalmente retorna null, potencialmente expondo detalhes de implementação interna ou causando comportamento inesperado explorável por atacantes.

\- Código Refatorado:

A vulnerabilidade de segurança relacionada a este código reside nas funções schemaPromise e schemaFactory, que podem retornar nulo -- comprometendo a integridade dos dados e expondo detalhes de implementação internos.

Aqui está o trecho de código refatorado:

```javascript
// ...resto do seu código permanece o mesmo

// Refatorar schemaPromise para rejeitar quando nulo for retornado
const schemaPromise = new Promise((resolve, reject) => {
const schema = createSchema({
typeDefs: /* GraphQL */ `
type Query {
foo: Boolean
}
`,
resolvers: {
Query: {
foo: () => true,
},
},
});

if (schema !== null && typeof schema !== "undefined") {
resolve(schema);
} else {
reject(Error("Falha na criação do esquema. Esquema nulo ou indefinido não é permitido."));
}
});

// Refatorar schemaFactory para lançar um erro quando nulo for retornado
const schemaFactory = () => {
const schema = // Código para criar o esquema
if (schema === null || typeof schema === "undefined") {
throw new Error("Falha na criação do esquema. Esquema nulo ou indefinido não é permitido.");
} else {
return schema;
}
};

// O restante dos seus casos de teste e código permanece o mesmo...
```

Nas modificações:

A função schemaPromise foi alterada para rejeitar e lançar um erro se o esquema for nulo ou indefinido. Fazendo isso, evitamos que a função retorne um esquema nulo ou indefinido e, assim, garantimos a integridade dos dados.

A função schemaFactory também foi refatorada para lançar um erro nos casos em que o esquema for nulo ou indefinido. Dessa forma, garantimos que a saída dessa função seja sempre a esperada e nunca nula ou indefinida.

Note que você pode modificar esses registros de erro para registrar informações sobre esses cenários e até mesmo interromper a execução, garantindo que a integridade do seu código seja mantida quando as circunstâncias não forem as esperadas.

Arquivo Envolvido: graphql-yoga/__tests__/schema.spec.ts

Linha de Código: 59

Titulo: CWE-XXXX - Registro Não Sanitizado Da Mensagem De Erro GraphQL No Caso De Teste 'O Fluxo De Eventos Com Erro Deve Ser Tratado (Erro Não GraphQL)'

Tipo de Scan: SAST

Severidade: Média

Nome do Componente: None

Versão do Componente: None

Como Mitigar: None

Descrição:

CWE-209, também conhecido como "Geração de Mensagem de Erro com Informação Sensível", descreve uma vulnerabilidade de segurança na qual o software, ao enfrentar condições excepcionais ou de erro, gera uma mensagem que inclui informações sensíveis sobre seu ambiente, usuários ou atividades associadas. Isso pode acontecer ao imprimir uma pilha de chamadas ou ao lançar um erro específico. O problema nesse caso é que, se a mensagem for exposta a um usuário não autorizado, ela pode revelar detalhes que auxiliam um atacante na realização de atividades maliciosas adicionais. A CWE-209 sugere que os desenvolvedores usem cuidado ao manipularemo mensagens de erro, garantindo que elas não revelem detalhes sensíveis.

Scanner: InspectAI

A mensagem de erro do GraphQL retornada pelo servidor não é sanitizada nem validada corretamente antes de ser registrada. Isso pode potencialmente permitir que informações sensíveis sejam registradas e divulgadas para atacantes.

\- Código Refatorado:

O problema está na parte de manipulação de erros do último caso de teste. Atualmente, ele registra todo o objeto GraphQLError em texto simples, o qual pode conter alguns dados sensíveis. Isso deve ser corrigido para evitar a exposição de dados nos logs. Apenas a mensagem de erro deve ser fornecida em vez do objeto GraphQLError completo. Aqui está o código refatorado:

```javascript
test('erro no fluxo de eventos deve ser tratado (erro não relacionado ao GraphQL)', async () => {
// ... código antes

const logging = {
debug: jest.fn(),
info: jest.fn(),
warn: jest.fn(),
// função de erro refatorada para registrar apenas a mensagem de erro,
// não o objeto GraphQLError completo
error: jest.fn().mockImplementation((error: GraphQLError) => {
```

```
console.error(`Ocorreu um erro: ${error.message}`);
}),
};

// ... código depois
});

test('erro no fluxo de eventos deve ser tratado (erro do GraphQL)', async () => {
// ... código antes

const logging = {
debug: jest.fn(),
info: jest.fn(),
warn: jest.fn(),
// função de erro refatorada para registrar apenas a mensagem de erro,
// não o objeto GraphQLError completo
error: jest.fn().mockImplementation((error: GraphQLError) => {
console.error(`Ocorreu um erro: ${error.message}`);
}),
};

// ... código depois
});
```

Nos blocos de código acima, a função de erro do registro (`logging`) registra apenas a mensagem de erro, e não o objeto GraphQLError completo, o qual pode conter dados sensíveis. Isso garante que informações sensíveis não sejam expostas por meio do registro de erros.

Arquivo Envolvido: graphql-yoga/__tests__/subscriptions.spec.ts

Linha de Código: 203

Titulo: CWE-xxx - Promessas De Esquema

Tipo de Scan: SAST

Severidade: Média

Nome do Componente: None

Versão do Componente: None

Como Mitigar: None

Descrição:

A CWE-754 é uma vulnerabilidade de segurança que se refere ao uso inadequado de funções criptográficas. Essa vulnerabilidade ocorre quando as funções criptográficas são usadas de maneira incorreta ou inadequada, o que pode comprometer a integridade e confidencialidade dos dados. Isso pode acontecer de várias maneiras, como usar algoritmos criptográficos fracos, armazenar chaves criptográficas de forma insegura, implementação imprópria de algoritmos criptográficos ou até mesmo não usar as funções criptográficas apropriadas para proteger os dados. Essa vulnerabilidade pode possibilitar a um atacante comprometer o sistema, decifrar informações sensíveis ou realizar ataques de força bruta para quebrar códigos criptografados. É essencial que os desenvolvedores utilizem corretamente as funções criptográficas e adotem boas práticas de segurança para evitar a exploração dessa vulnerabilidade.

Scanner: InspectAI

O código cria uma promessa que resolve para um esquema, porém não trata quaisquer erros potenciais que possam ocorrer durante a resolução da promessa, resultando em exceções não tratadas e potenciais vulnerabilidades de segurança.

\- Código Refatorado:

Para lidar com possíveis erros ao resolver promessas, adicionaremos blocos try-catch para as operações envolvidas com promessas. O tratamento de erros é importante para evitar falhas inesperadas devido a exceções não tratadas.

Código fonte atualizado:

```jsx
// Outros imports
import { GraphQLSchema } from 'graphql';
import { createSchema, createYoga } from '../src/index.js';
describe('schema', () => {

// Código existente

it('promessa do schema', async () => {
const yoga = createYoga({
```

```
schema(async() => {
try {
return Promise.resolve(
createSchema({
typeDefs: /* GraphQL */ `
type Query {
foo: Boolean
}
`,
resolvers: {
Query: {
foo: () => true,
},
},
}),
);
} catch (error) {
// Adicionar tratamento adequado de erros
console.log(error);
}

}),
});
const query = /* GraphQL */ `
query {
foo
}
`;
const result = await yoga.fetch('http://yoga/graphql', {
method: 'POST',
body: JSON.stringify({ query }),
headers: {
'Content-Type': 'application/json',
},
});
```

```
const { data } = await result.json();

expect(data).toEqual({

foo: true,

});

});

// O restante dos testes pode ser modificado de forma semelhante.

});
```

A partir da descrição da vulnerabilidade relatada, o problema principal parece ser a falta de tratamento de erros nas Promises. Envolver o código que pode lançar exceções com um bloco try-catch é uma prática recomendada para lidar com possíveis erros. O código refatorado acima agora trata as possíveis rejeições dentro da Promise, capturando o erro e tratando-o adequadamente. O restante dos testes existentes pode ser tratado de forma semelhante, adicionando blocos try-catch para lidar com possíveis rejeições dentro das Promises.

Arquivo Envolvido: graphql-yoga/__tests__/schema.spec.ts

Linha de Código: 74
