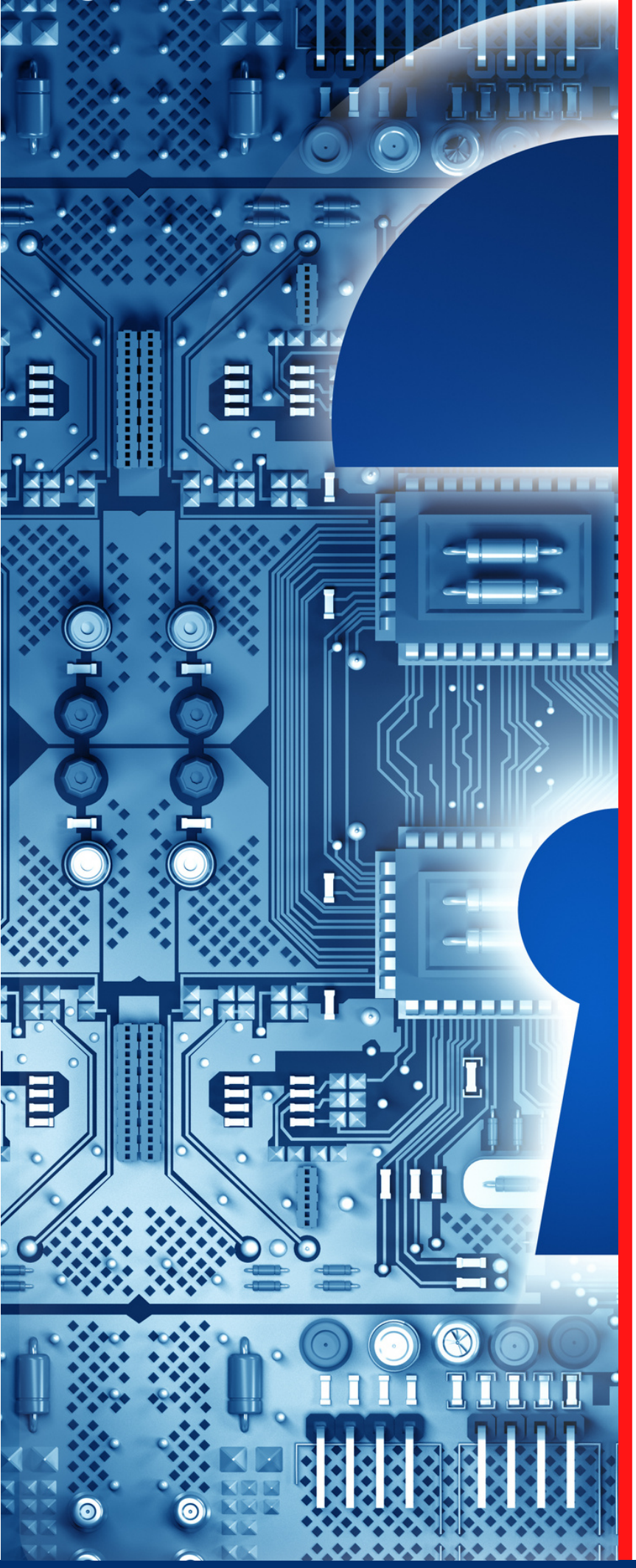


OSIC

ORIENTAÇÃO DE SEGURANÇA DA INFORMAÇÃO E CIBERNÉTICA

06/2023

Vazamento de Dados em Aplicativos *Web*

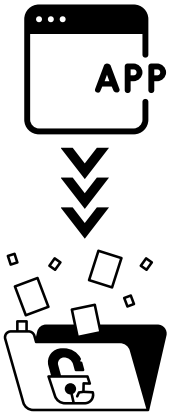
Vulnerabilidades e Ataques

Textos: João Alberto Muniz Gaspar
Diagramação: Douglas Rocha de Oliveira
Produção: Secretaria de Segurança da Informação e Cibernética

TLP:CLEAR

Vazamento de dados em aplicativos *web*

O vazamento de dados (*Data Leaking* ou *Data Spilling*), também intitulado como divulgação não autorizada de informações (*information disclosure*), ocorre, em muitos casos, quando ocorre falha em aplicativo *web*, expondo, para agentes não autorizados, informações sensíveis e confidenciais.



Ressalta-se que, apesar do vazamento de dados geralmente ser fruto de interação maliciosa ou inesperada com o aplicativo *web*, ocasionalmente, este incidente pode ocorrer de forma não intencional, por falhas de configuração, para usuários que estão simplesmente navegando normalmente no *site*.

Dependendo do contexto, os *sites* podem vazar diversos tipos de informação para um *hacker* em potencial, incluindo:

1. **dados sobre usuários, como nomes ou informações financeiras;**
2. **dados confidenciais comerciais ou industriais; e**
3. **detalhes técnicos sobre o *site* ou sua infraestrutura.**

Os vazamentos de dados podem surgir de inúmeras maneiras, mas, geralmente, podem ser categorizados da seguinte forma:

1. **falha ao remover o conteúdo interno do conteúdo público;**
2. **configuração insegura do *site* ou de tecnologias relacionadas; e**
3. **projeto e comportamento falhos do aplicativo.**

Impactos do Vazamento de Dados

As vulnerabilidades de vazamento de dados podem ter impacto direto ou indireto, dependendo da finalidade do *site* e, portanto, de quais informações um adversário possa obter. Apenas o ato de divulgar informações confidenciais pode gerar um alto impacto para as partes afetadas.

No geral, os administradores de *sites* estão esclarecidos sobre o alto risco do vazamento de dados de usuários ou de negócios, pois as consequências estão bastante claras na legislação correlata; em capacitações sobre o tema; e nos programas de certificação profissional em segurança e privacidade de dados. Os dados pessoais estão sempre em destaque devido a Lei Geral de Proteção de Dados Pessoais (LGPD). No entanto, a divulgação não autorizada de informações técnicas sobre o próprio *site* pode ter efeitos graves de altíssimo impacto.

No tocante às informações técnicas, como a estrutura de diretórios ou *frameworks*, mesmo de uso limitado, estas são devem ser observadas na análise de risco de aplicativos da *web*, pois podem ser usadas no ciclo de vida de um ataque. Um *hacker* poderá utilizar da informação técnica, obtida em um vazamento de dados, como um estágio inicial para expor outras vulnerabilidades que sejam interessantes para o atacante, em especial, quando a informação permite implementar ataques mais complexos ou de alto impacto em alvos considerados vantajosos.

A severidade do vazamento de dados dependerá do que o *hacker* pode realizar com a informação obtida. O incidente deve ser considerado um problema de alta gravidade sempre que o *hacker* puder realizar uma atividade com prejuízo à instituição.

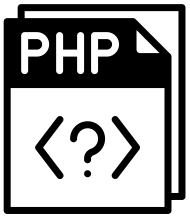
Por exemplo, o conhecimento de que um *site* está usando uma versão específica da estrutura é de uso limitado, caso esta versão esteja atualizada. Entretanto, esse tipo de informação torna-se significativa quando o *site* está usando uma versão não atualizada que contenha vulnerabilidades conhecidas. Em ambientes expostos, executar um ataque severo ganha menor complexidade ao executor, pois, por vezes, a exploração da vulnerabilidade identificada possui vasta documentação pública disponível.

Infere-se sobre a importância, ao se identificar que houve um vazamento, em identificar as ações utilizadas pelo atacante, assim como as consequências do uso do dado vazado. Geralmente, pequenos detalhes técnicos podem ser descobertos de várias maneiras em muitos dos *sites* que são alvos. Dessa forma, ao analisar um incidente desse tipo, o foco principal deve estar no impacto e na capacidade de exploração das informações vazadas, e não apenas na presença da divulgação não autorizada de informações. A exceção óbvia a isso é quando a informação vazada é tão sensível que merece atenção por si só.

Exemplos de Vulnerabilidades e Ataques de Vazamento de Dados

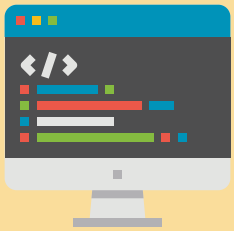
1) Captura de *Banner* (*Banner Grabbing* ou Reconhecimento Ativo)

Ataque durante o qual o *hacker* envia solicitações, de forma manual ou usando ferramentas OSINT[1] (*Open Source Intelligence*), ao sistema que seja de seu interesse para coletar mais informações sobre ele. Se o sistema não estiver configurado com segurança, ele pode vazar informações sobre si mesmo, como a versão do servidor, versão do seu código de programação PHP/ASP.NET, versão dos utilitários de segurança *Open*SSH, etc.



Na maioria dos casos, a captura de *banner* não envolve o vazamento de dados críticos, mas, sim, de informações que podem ajudar o *hacker*. Por exemplo, se o alvo vazar a versão do PHP em execução no servidor e essa versão for vulnerável à execução remota de código (RCE, na sigla em inglês), o *hacker* poderá explorar essa vulnerabilidade e até assumir o controle do aplicativo *web*.

2) Divulgação Não Autorizada de Código-Fonte (*Source Code Disclosure*)

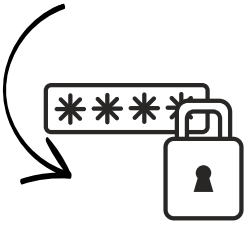


Problemas de divulgação não autorizada do código-fonte ocorrem quando o código do ambiente de *backend* de um aplicativo da *web* é exposto ao público. A divulgação não autorizada desse código permite que um *hacker* entenda como o aplicativo se comporta e verifique se há falhas lógicas ou informação sensível incluída diretamente no código-fonte (*hardcoded*).

A gravidade depende de quanto do código-fonte é exposto e de quão críticas para a segurança do aplicativo *web* são as suas linhas vazadas. Resumindo, a divulgação não autorizada desse código transforma um processo de 'teste de caixa preta' em uma abordagem de 'teste de caixa branca', pois os *hackers* têm acesso completo a ele. Problemas de divulgação não autorizada de código-fonte podem ocorrer de várias maneiras. Entre elas, destacam-se as seguintes:

- repositório de código-fonte público desprotegido: uma das formas de melhorar os métodos de desenvolvimento colaborativo é hospedar o código-fonte em um repositório na nuvem. Várias empresas que desenvolvem *software* de código aberto usam repositórios públicos para que o público possa contribuir com o projeto. No entanto, muitos desses repositórios não possuem mecanismos fortes de proteção, o que pode permitir que *hackers* acessem o código-fonte e as informações hospedadas; e

- informação sensível *hardcoded*: outro problema comum é a existência de informações sensíveis ou confidenciais codificadas diretamente no código-fonte, como credenciais de usuário ou chaves secretas de API. Um *hacker* pode facilmente usar essas informações para sabotar serviços ou causar problemas de acesso para usuários legítimos. Além disso, outras informações que normalmente são encontrada *hardcoded* em ataques de vazamento de dados são os endereços IP internos, o que permite ao *hacker* identificar e aprender sobre a topologia da rede interna. Esse conhecimento pode ser usado para navegar por essa rede e atacar vários sistemas por meio de um *server-side request forgery* (SSRF)[2], por exemplo.

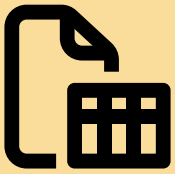


3) Uso incorreto do Tipo MIME

O tipo MIME (MIME *Type*) é o mecanismo utilizado para dizer ao cliente a variedade de documentos transmitidos. É importante que o servidor esteja configurado adequadamente, de modo que o tipo MIME correto seja transmitido com cada documento.

Os navegadores da *web* costumam usar o tipo MIME para determinar qual ação usar como padrão para ser executada quando um recurso é obtido. Eles sabem como analisar as informações que recebem graças ao cabeçalho *Content-Type* HTTP que o servidor da *web* envia em suas respostas HTTP.

Em caso de ausência de um tipo MIME, má-configuração dele, ou em alguns casos e circunstâncias específicos de acordo com cada navegador, pode ocorrer um MIME *sniffing*.

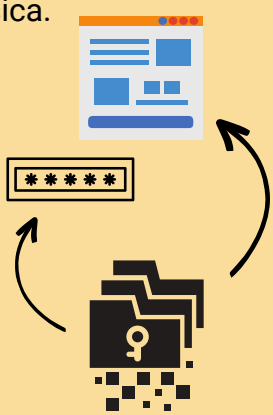


O MIME *sniffing* pode ser definido como a prática adotada pelos navegadores para determinar o tipo MIME efetivo de um recurso da *web* examinando o conteúdo da resposta em vez de confiar no cabeçalho *Content-Type*. Cada navegador executa isso de forma diferente e em circunstâncias diferentes.

Embora o MIME *sniffing* possa ser útil para determinar o formato de arquivo correto de um recurso, ele também pode causar uma vulnerabilidade de segurança. Essa vulnerabilidade pode ser bastante perigosa tanto para os proprietários do *site* quanto para seus visitantes. Isso ocorre porque um *hacker* pode aproveitar a detecção de MIME para enviar um ataque XSS (*Cross Site Scripting*).[3]

4) Vazamento de Nomes e Caminhos de Arquivos

Em razão do tratamento inadequado da entrada do cliente, casos especiais de *backend* ou configurações inadequadas do servidor da *web*, um aplicativo *web* pode revelar nomes de arquivos, registros e diretórios, o que fornece conhecimento sobre o *design* da estrutura básica.



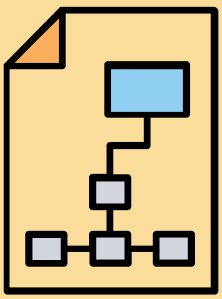
Hackers utilizam ataques do tipo *path traversal* para realizar esse tipo de operação. Ataques desse tipo forçam o acesso a arquivos, diretórios e comandos localizados fora do diretório raiz do documento da *web* ou do diretório raiz CGI. Para isso, o *hacker* pode explorar uma URL de forma que o *site* da *web* execute ou divulgue o conteúdo dos arquivos no servidor da *web*. Embora a maioria desses *sites* restrinja o acesso do usuário à raiz do documento da *web* ou ao diretório raiz CGI (*Common Gateway Interface*), um *hacker* pode obter acesso a esses diretórios usando sequências de caracteres especiais.

A sequência ../ é uma das mais comumente usadas por invasores para acessar arquivos ou executar comandos no sistema de arquivos. Outras sequências muito utilizadas em ataques de *path traversal* são:

- codificação Unicode válida e inválida ..%u2216 ou ..%c0%af do caractere de barra;
- caracteres de barra invertida ..\ em servidores baseados em Windows;
- caracteres codificados de URL, como %2e%2e%2f; e
- codificação de URL dupla ..%255c do caractere de barra invertida.

Esse ataque pode ser potencializado utilizando um código malicioso externo injetado diretamente como variável na aplicação, sendo neste caso conhecido também como RFI (*Remote File Injection*).[4]

5) Listagem de diretório

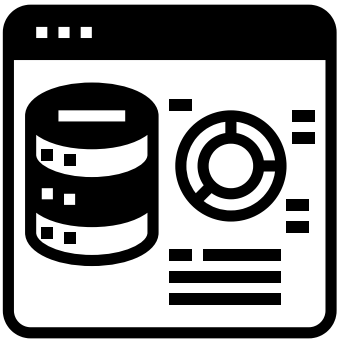


A listagem de diretórios em servidores da *web* pode causar o vazamento de nomes e caminhos de arquivos. Essa funcionalidade é fornecida por padrão em servidores da *web*. Com ela, quando não há página padrão para mostrar, o servidor retorna uma lista de arquivos e diretórios presentes no *site*.

Por exemplo, se o nome de arquivo padrão em um servidor da *web* Apache for index.php e não houver sido carregado um arquivo chamado index.php no diretório raiz, o servidor mostrará uma lista de diretórios do diretório raiz em vez de analisar o arquivo PHP.

Deixar tal funcionalidade habilitada em ambientes de produção é uma prática ruim e pode levar a problemas maiores de segurança, como um ataque de *directory indexing*. Esse tipo de ataque explora a função do servidor *web* que lista todos os arquivos dentro de um diretório solicitado se o arquivo base normal não estiver presente.

Ataques do tipo *directory indexing* podem ocorrer pela exploração das seguintes vulnerabilidades:



- servidor *web* configurado incorretamente para permitir ou fornecer um índice de diretório;
- o servidor *web* permite um índice de diretório mesmo que tenha sido desabilitado no arquivo de configuração, ou uma página de índice está presente; e
- o banco de dados de cache usado pelo Google contém dados históricos, incluindo índices de diretório de varreduras anteriores de um *site* específico.

Atualmente, a maioria dos administradores *web* estão cientes de que tal funcionalidade deve ser desativada, de tal forma que esse problema já não é tão comum.

Melhores Práticas para a Mitigação das Vulnerabilidades de Vazamento de Informações

1. Identificar as informações consideradas sensíveis e providenciar que todos os envolvidos na produção de um *site* estejam totalmente cientes da importância de as manter seguras. Por vezes, informações aparentemente inofensivas podem ser muito mais úteis para um *hacker* do que as pessoas imaginam. Destacar esses perigos pode ajudar a garantir que as informações sensíveis, em geral, sejam tratadas com mais segurança pela sua organização.

2. Configurar e testar o servidor da *web* para:
   - não permitir a listagem de diretórios e certificar-se de que o aplicativo da *web* sempre mostre uma página padrão;
   - não enviar cabeçalhos de reação ou informações básicas que revelem detalhes sobre o *site* ou a infraestrutura disponível;
   - garantir que todas as exceções sejam tratadas quando o aplicativo da *web* falhar e nenhuma informação técnica for relatada nos erros, retornando, preferencialmente, uma página de erro genérica; e
   - utilizar, sempre que possível, mensagens de erro genéricas.

3. Configurar os serviços em execução nas portas abertas do servidor de forma a evitar que revelem informações sobre as suas construções e versões.

4. Garantir que os controles e credenciais de acesso adequados estejam em vigor em todos os servidores, serviços e aplicativos.

5 Empregar validações suficientes no código de *backend* para capturar todas as exceções e evitar o vazamento de dados.

6 Realizar um processo de auditoria de forma a não permitir, em qualquer aplicativo *web*, a existência *hardcoded* de informações sensíveis, como *login*/senha, chaves API, endereços IP ou dados pessoais, nem mesmo nos comentários do código.

7 Configurar os tipos MIME apropriados para cada um dos documentos utilizados pelos nos aplicativos *web* nos servidores da organização para evitar ataques causados por MIME *sniffing*.

- Definir o cabeçalho de resposta HTTP *X-Content-Type-Options*[5] como *nosniff* instrui os navegadores a desabilitar o conteúdo ou o MIME *sniffing*. Isso obrigará os navegadores que suportam detecção de MIME a usar o tipo de conteúdo fornecido pelo servidor e não interpretar o conteúdo como um tipo de conteúdo diferente. O HTTP *X-Content-Type-Options* é suportado por Chrome, Firefox e Edge, assim como outros *browsers*.
- Como método alternativo para evitar ataques XSS devido ao MIME *sniffing*, pode-se utilizar um subdomínio separado para hospedar e entregar todo o conteúdo carregado pelo usuário. Isso ajuda a garantir que nenhum conteúdo carregado entre em contato com o domínio principal da *web*.

8 Sempre que possível, não transferir material considerado sensível ou reservado para um servidor *web*, limitando-se apenas à informação que seja estritamente necessária.

9 Empregar validações suficientes no código de *backend* para capturar todas as exceções e evitar o vazamento de dados.

10 Verificar continuamente se cada uma das solicitações para criar/editar/visualizar/excluir recursos possui controles de acesso adequados de forma a evitar problemas de escalonamento de privilégios e garantir que todas as informações sensíveis permaneçam protegidas.

O Departamento de Segurança da Informação e Cibernética (DSIC) recomenda aos integrantes da Administração Pública Federal, observar o previsto no Plano de Gestão de Incidentes Cibernéticos, particularmente quanto a prevenção, disponível em:

https://www.gov.br/gsi/pt-br/composicao/SSIC/dsic/plano-de-gestao-de-incidentes-ciberneticos-plangic/plangic.pdf

O DSIC orienta que qualquer usuário ao identificar um incidente cibernético, como o vazamento de dados em aplicativos web, informe imediatamente a Equipe de Prevenção, Tratamento e Resposta a Incidentes Cibernéticos (ETIR) da instituição.

Outros conceitos podem ser verificados no glossário disponível em:

https://www.gov.br/gsi/pt-br/assuntos/dsi/glossario-de-seguranca-da-informacao-1

Por fim, o DSIC solicita, ainda, que propostas de temas, sugestões ou outras contribuições sejam encaminhadas ao e-mail educa.si@presidencia.gov.br para fomentar futuras emissões da OSIC.

TLP:CLEAR

Informações complementares

1. **Ferramentas OSINT:** o termo OSINT significa "*Open Source Intelligence*", que se refere ao uso de informações disponíveis publicamente para fins de inteligência e investigação. Essas ferramentas podem ser usadas para uma variedade de propósitos, como pesquisa de mercado, análise de concorrência, investigação de fraudes, análise de ameaças à segurança, entre outros.

2. ***Server-side request forgery* (SSRF):** é uma vulnerabilidade de segurança que ocorre quando um atacante consegue enganar um servidor para que ele faça solicitações a outros sistemas externos, sem o conhecimento ou consentimento do proprietário do servidor. Isso pode permitir que o atacante acesse dados confidenciais, execute comandos maliciosos ou comprometa a segurança do sistema.

3. **Ataque XSS (*Cross Site Scripting*):** é um tipo de ataque de segurança que ocorre quando um invasor injeta código malicioso em uma página da *web* acessada por outros usuários. Esses códigos maliciosos são executados no navegador dos usuários, permitindo que o atacante roube informações confidenciais, como senhas e *cookies*, ou execute ações maliciosas em nome do usuário.

4. **RFI (*Remote File Injection*):** é uma vulnerabilidade de segurança que ocorre quando um invasor consegue injetar um código malicioso em um *site* ou aplicação da *web* para executar arquivos remotos hospedados em servidores controlados pelo invasor. Isso pode permitir que o atacante execute comandos maliciosos, roube informações confidenciais ou comprometa a segurança do sistema

5. **Cabeçalho de resposta HTTP *X-Content-Type-Options*:** é uma diretiva de segurança que pode ser incluída em uma resposta HTTP para instruir um navegador da *web* a não realizar a detecção automática do tipo MIME do conteúdo. Isso ajuda a prevenir ataques de tipo MIME *sniffing*, nos quais um invasor pode tentar explorar a maneira como um navegador interpreta o tipo de arquivo de uma resposta HTTP para executar código malicioso.