



Segurança Web: Modelo

SOP (Same Origin Policy)

1) To understand what might happen without the SOP: <https://www.netsparker.com/whitepaper-same-origin-policy/#WorldWithoutSameOriginPolicy>.

Basicamente, podemos ser enganados a aceder um website com url: `www.badbank.com` o qual cria um `iframe` do `www.bank.com`. Com isto, podemos pensar que estamos no site legítimo e por isso, fazemos login no site. Dessa forma o badsite pode recolher nossas informações ou até mesmo enviar requests para o nosso banco.

2) Origin: `host, protocol, port`.

3) The Same Origin Policy não apenas evita com que possamos fazer requests a outros origens cegamente.

- **Escrita:** É sempre possível escrever: podemos enviar/expor dados.
- **Leitura:** A leitura, no entanto, possui algumas condições.

Leitura

- **HTML:** Não podemos analisar o código programaticamente e podemos criar frames. Um exemplo disso são as imagens: não podemos ver quantos pixels uma image tem, mas podemos ver o seu tamanho.
- **Javascript:** Pode ser **executado**, mas somente no nosso contexto e **um código de javascript não pode alterar outro javascript de uma origem diferente**. Além disso, **não podemos analisar o código programaticamente**.
- **Iframe:** Assim como as imagens, não podemos ler o conteúdo do Iframe programaticamente nem um iframe pode aceder os recursos da página que está a exibir. É um sistema isolado. Melhor explicação:
<https://security.stackexchange.com/questions/67889/why-do-browsers-enforce-the-same-origin-security-policy-on-iframes>

[Slide 21~25]

Mais sobre o SOP:

CORS (Cross-Origin Resource Sharing):

Assim como estabelecido no **SOP**, podemos fazer requests para qualquer site, mas é de **responsabilidade do servidor verificar se esse pedido é aceito ou não**.

O browser basicamente verifica se o pedido de um cliente **A** pode aceder a um recurso de **B**.

O servidor pode permitir mais casos de uso através do atributo **Access-Control-Allow-Origin**.


Pre-flight request

Antes de fazer qualquer pedido, o cliente que fez o request precisa enviar um **dummy request** ao servidor, a fim de verificar se ele **também implementa o CORS** e se um **website A pode aceder aos recursos de um website B**.

Qual a motivação de um pre-flight request

What is the motivation behind the introduction of preflight CORS requests?

Preflight requests were introduced so that a browser could be sure it was dealing with a CORS-aware server before sending certain requests. Those requests were defined to be those that were both potentially dangerous (state-changing) and new

 <https://stackoverflow.com/questions/15381105/what-is-the-motivation-behind-the-introduction-of-preflight-cors-requests>



Without the preflight request, servers could begin seeing unexpected requests from browsers. This could lead to a security issue if the servers weren't prepared for these types of requests. The CORS preflight allows cross-domain requests to be introduced to the web in a safe manner.

Com SOP não seria possível que **websites de outras origens** enviassem requests apropriados a um website A (devido a falta de cookies e outros recursos para verificar a autenticidade).

Como o CORS permite o relaxamento de políticas de SOP. Vamos voltar ao exemplo do www.bank.com. Vamos supor que www.badbank.com tenha políticas relaxadas de SOP com o CORS e que www.bank.com não implemente **CORS**. No fim, www.bank.com, por ser um website criado antes da existência do **CORS** não está a espera de um pedido de outro recurso, por isso o www.badbank.com poderia enviar um request para deletar a conta ou transferir dinheiro.

Por isso, o **pre-flight** foi feito, para poder suportar a transição a web sem **CORS** para a web com **CORS** de maneira segura.

Com o **pre-flight**, www.badbank.com iria enviar um **dummy request** ao www.bank.com pergutando se ele tem **CORS** implementado, se não tiver, o pedido é rejeitado. E mesmo que tivesse o **CORS** o pedido iria ser rejeitado, porque www.badbank.com não seria um website com permissões de acesso.

[slide 26~27]

Cookies

Origem

Definido por: **path**, **dominio**. O esquema é opcional ao contrário dos requests.

Public suffix list

É basicamente uma lista de sufixos que são comuns a quase todas a páginas como: **.com**, **.edu**, ...

Para quem enviar cookies

Tradicionalmente os browsers enviam as cookies no pedido de uma url, mas somente se a cookie tiver a mesma origem do website:

- Mesmo domínio da url.
- E se a path da cookie tiver o domínio for um prefixo da path da url.

	Do we send the cookie?		
Request to URL	Set-Cookie: ...; Domain=login.site.com; Path=/;	Set-Cookie: ...; Domain=site.com; Path=/;	Set-Cookie: ...; Domain=site.com; Path=/my/home;
checkout.site.com	No	Yes	No
login.site.com	Yes	Yes	No
login.site.com/my/home	Yes	Yes	Yes
site.com/my	No	Yes	No

SOP para cookies

Os browsers possuem opções a dizer para quem deve-se enviar a cookie:

`SameSite=Lax` OU `SameSite=Strict`.

A opção `Strict` envia cookies apenas para websites com a **mesma origem que a top level frame**. Portanto, o browser não iria enviar cookies do `www.bank.com` para o `www.badbank.com`, já que ambos possuem top level frames diferentes.

A opção `Lax`, no entanto, abre algumas exceções.

Ainda, um javascript que execute dentro da página de um banco (e.g google analytics) poderia aceder as cookies do cliente usado `document.cookie` (session Hijacking). Mas podemos evitar isto adicionando `HTTPOnly` ao cookie.

Perguntas

1) O que é o SOP? Quais são as limitações que este impõe para HTTP, HTML e Javascript?

SOP (Same Origin Policy) é uma política de proteção.

Sempre podemos escrever pedidos HTTP. O conteúdo escrito na resposta não pode ser visualizado programaticamente, no entanto pode ser processado. Para o HTML e Javascript temos a mesma coisa: podemos ler e executar o conteúdo, mas não podemos analisá-lo programaticamente.

2) O que é considerado um recurso da mesma origem?

Tem de possuir o mesmo esquema, nome de domínio e porta.

3) A CIA são conceitos de segurança usados amplamente, a fim de identificar os tipos de segurança que um recurso define. A SOP possui aplica quais destes conceitos?

SOP tenta garantir confidencialidade (pelo que não é possível analisar programaticamente dados vindo de outras origens). Também garante integridade, pelo que não é possível alterar dados vindos de outras origens.

4) O que é o CORS?

É o cross-origin resource sharing. Permite relaxar as políticas do SOP por meio da abertura de algumas exceções à política.

5) O que são pedidos pre-flight?

É um dummy request feito, a fim de estabelecer a comunicação. Um cliente (lado A) envia este dummy request ao (lado B) e este envia uma resposta permitindo o acesso ou negando-o.

6) Qual a definição de origem para um cookie?

É apenas o domínio e path.

Ou seja, enquanto o URL tem o esquema, domínio e porta, o cookie tem o domínio e path.

7) O SOP permite com que o browser envie cookies para quais domínios?

Um cookie pode ser enviado se a cookie for um sufixo do domínio da url e se a path da cookie for um prefixo da path da URL.

8) Explique o SameSite=Strict e SameSite=Lax?

9) Como podemos evitar que um cookie seja lido por javascript?

Colocando a flag HTTPOnly=true

10) Complete o esquema abaixo:

	Do we send the cookie?		
Request to URL	Set-Cookie: ...; Domain=login.site.com; Path=/;	Set-Cookie: ...; Domain=site.com; Path=/;	Set-Cookie: ...; Domain=site.com; Path=/my/home;
checkout.site.com			
login.site.com			
login.site.com/my/home			
site.com/my			

O website tem de estar “contido” no url.

11) Por quê utilizar (i)frames?