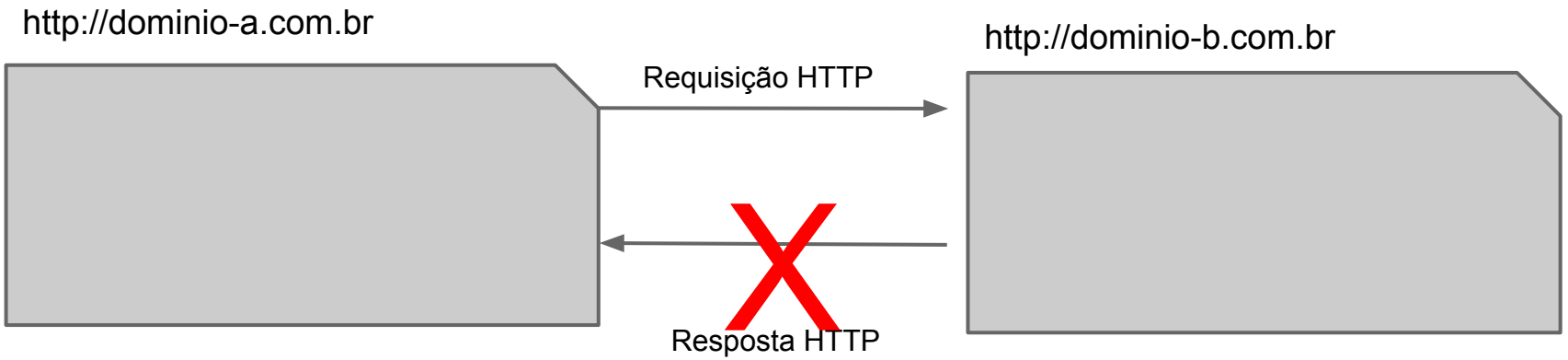
Como NÃO se deve utilizar o CORS (HTML 5 Security)

FLARE SECURITY

Securing People, Process and Technology

Same Origin Policy

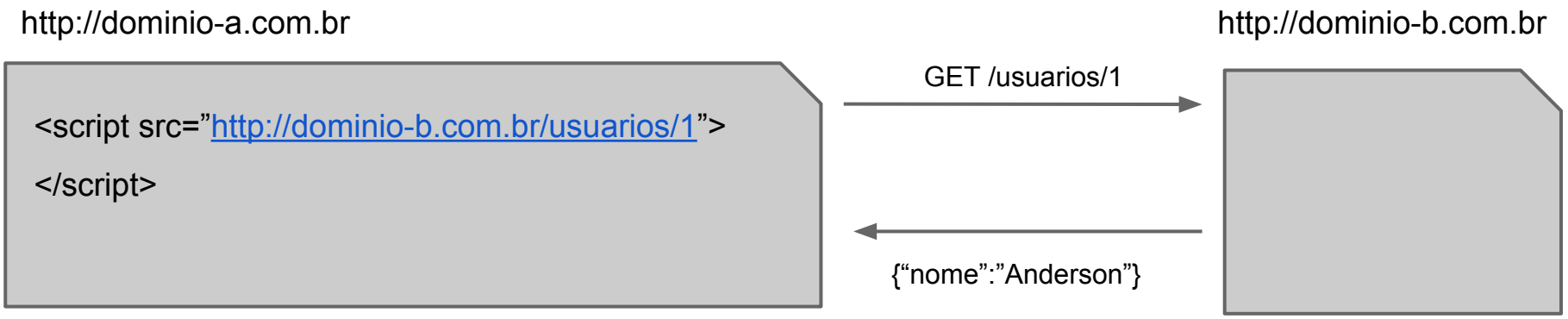


Por definição, nenhuma página de um domínio consegue ter acesso à resposta de requisições (inclusive ajax) para outros domínios, salvo algumas exceções elencadas na Same Origin Policy, uma especificação da W3C.

JSON-P
a solução paliativa (1-2)

Uma das poucas tags permitidas pela Same Origin Policy a realizar requisições entre domínios é a tag **<script>**, que foi explorada pelo desenvolvedores para burlar o conceito da Same Origin Policy, porém dando margem à problemas de segurança.

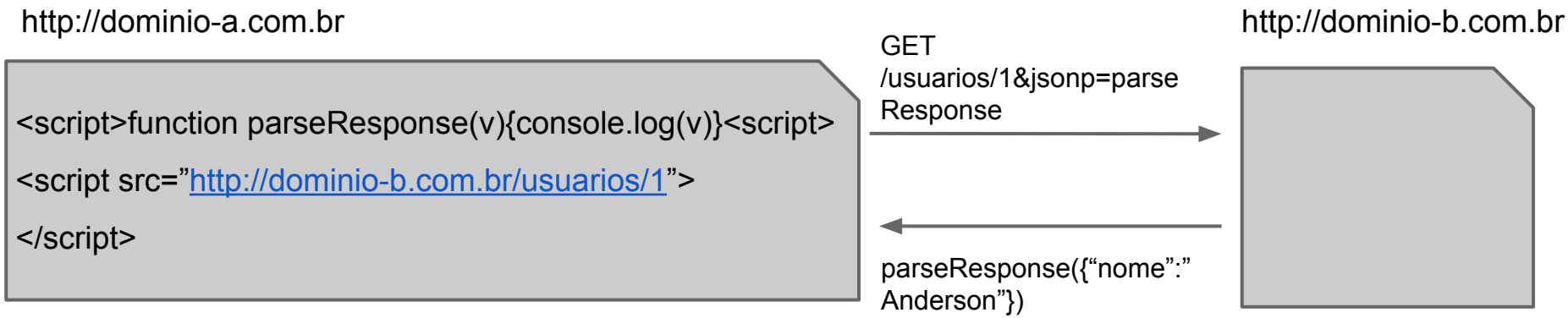
Sem JSON-P
Requisição a um web service simples



Não há como manipular a resposta nesse exemplo porque não é possível atribuir a nenhuma variável.

JSON-P
a solução paliativa (2-2)

Com JSON-P
Requisição a um web service simples

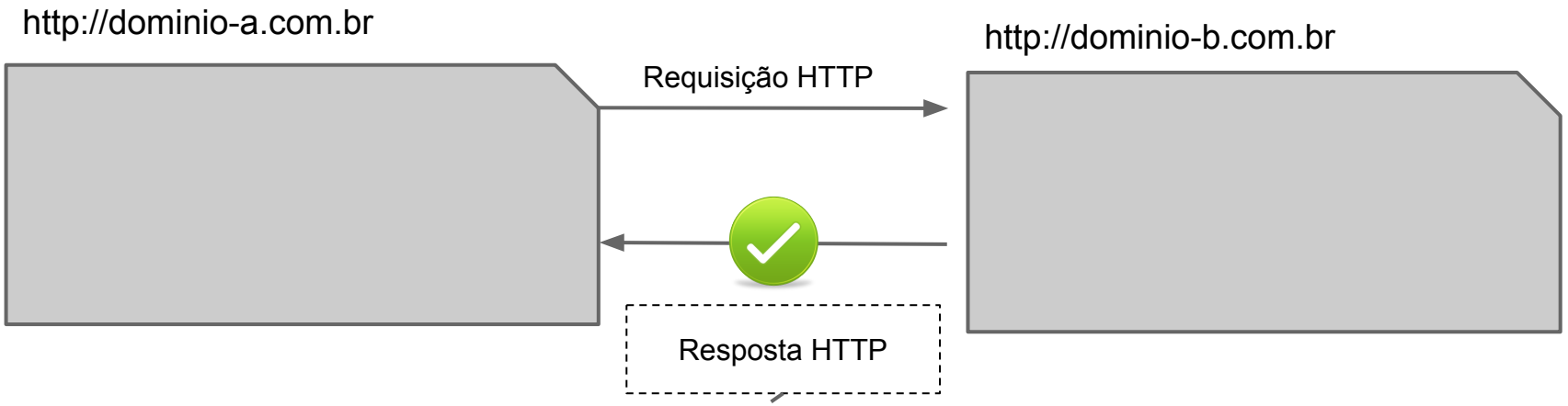


Principais problemas da perspectiva de segurança:

1) Dominio-a.com.br pode ser comprometido se o dominio-b.com.br resolver retornar um javascript malicioso, visto que json-p retorna código arbitrário e não apenas json;

2) Os web services providos pelo dominio-b ficam disponíveis não apenas ao domínio-a, mas sim para todos os websites,

Cross-Origin Resource Sharing (CORS) (1-3)



Com a especificação CORS é possível autenticar e autorizar domínios de modo granular por página.

Neste caso, bastou o dominio-b adicionar o cabeçalho de resposta ao lado para permitir que o dominio-a realize requisições sem se preocupar com a Same Origin Policy.

Cross-Origin Resource Sharing (CORS) (2-3)

Essa especificação permite filtrar métodos HTTP, filtrar cabeçalhos customizados enviados na resposta e muito mais.

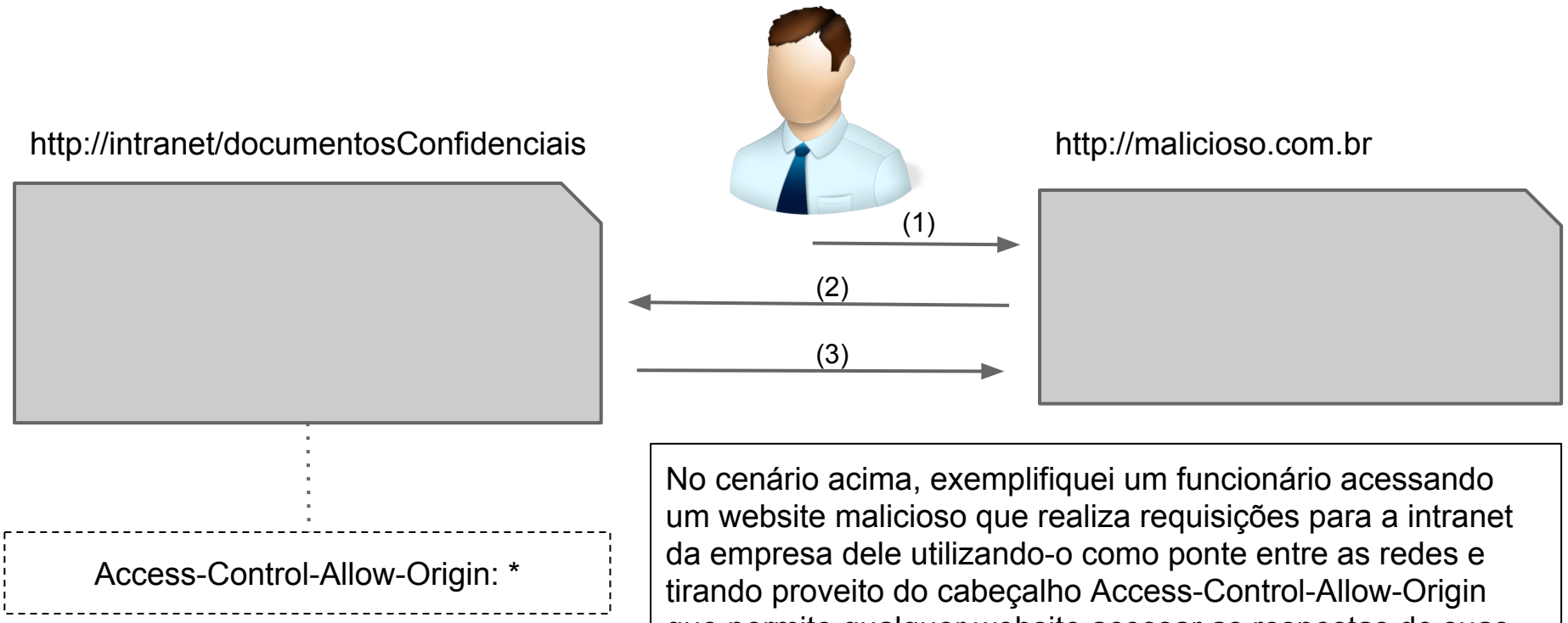
Há também o conceito de Preflight Request, uma requisição de confirmação enviada pelo navegador antes da requisição desejada ser disparada, mas não vou abordá-la nessa apresentação.

A grande questão aqui é que muitos desenvolvedores se deparam com a Same Origin Policy e na ansiedade de fazer suas aplicações funcionarem acabam adicionando o cabeçalho:

```
Access-Control-Allow-Origin: *
```

Exemplo clássico de programadores orientados ao site Stack Overflow: http://stackoverflow.com/questions/10143093/origin-is-not-allowed-by-access-control-allow-origin

Cross-Origin Resource Sharing (CORS) (3-3)



No cenário acima, exemplifiquei um funcionário acessando um website malicioso que realiza requisições para a intranet da empresa dele utilizando-o como ponte entre as redes e tirando proveito do cabeçalho Access-Control-Allow-Origin que permite qualquer website acessar as respostas de suas requisições, resultando assim em vazamento de informações confidenciais da empresa.

Conclusão

Com base nos riscos associados à liberação de diversos domínios por meio do cabeçalho Access-Control-Allow-Origin: *, a utilização do CORS deve ser realizada de maneira ponderada e granular seguindo o princípio do Privilégio Mínimo, evitando assim a exposição desnecessária e reduzindo a superfície de ataque.

Referências

- http://www.html5rocks.com/en/tutorials/cors/
- http://www.w3.org/TR/cors/
- http://en.wikipedia.org/wiki/Same_origin_policy
- http://arunranga.com/examples/access-control/
- http://en.wikipedia.org/wiki/JSONP

Obrigado

www.flaresecurity.com

Anderson Dadario