UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Luana Paula Carla da Silva

Problemas de Segurança em Aplicações Cross-domain: Algumas formas de Detecção e Tratativa

Uberlândia, Brasil

2021

Resumo

A evolução tecnológica vivenciada atualmente - tanto em relação aos softwares quanto aos hardwares mais populares – que abastecem o mercado também ocorre em relação às ameaças no assim chamado mundo virtual. Aliado ao fato do avanço tecnológico, existe, também, a popularização do acesso a Internet por banda larga, que possibilita uma major velocidade na proliferação de ameaças. No entanto, os mecanismos de segurança da informação nem sempre estão aptos a detê-las, sendo, portanto, necessário desenvolver um projeto de segurança adequado aos negócios e características de cada empresa, pois um conhecimento específico das vulnerabilidades a um passo fundamental para se minimizar os efeitos de qualquer eventual ameaça. Em vista desse cenário, este trabalho apresenta uma solução para prover segurança da informação em organizações que utilizam sistemas em múltiplos domínios ou *cross domain*.

Palavras-chave: Segurança da informação, domínios, *cross domain*.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Atualmente, é consenso geral que se deve estabelecer a segurança da informação no meio organizacional para contrapor os constantes ataques e também para transmitir uma imagem de confiabilidade aos parceiros e clientes (LAUDON, 2003). Este quadro é agravado pelo crescimento exponencial da web e do volume de informações transacionado nos últimos anos. Para assegurar a segurança necessária, é fundamental que os mecanismos de gerenciamento de informações corporativas sejam capazes de reagir prontamente ao aparecimento de novas ameaças que surgem em função do ritmo acelerado das mudanças tecnológicas.

Neste contexto, o objetivo deste trabalho é melhoria da segurança da informação em um ambiente corporativo altamente dependente de sistemas computacionais; o que é plenamente justificável uma vez que s sociedade atual vive em plena "Era da Informação" (ALBERTIN; MOURA, 2001). Neste contexto, é essencial que as empresas invistam vultosos recursos humanos e financeiros substanciais que permitam proteger as informações associadas aos seus negócios. A necessidade de proteção é devida por várias razões, dentre elas: a existência de segredos industriais próprios e/ou de parceiros que devem ser tratados de maneira confidencial, já que seu vazamento pode comprometer a imagem dos negócios.

A implantação de sistemas que viabilizam ou apoiam práticas de manipulação segura da informação acarretam sensíveis transformações culturais e organizacionais no cotidiano da empresa. Soluções diversas estão a disposição e são executadas, visando a melhoria da competitividade da empresa no seu respectivo mercado. Além disso, todos os sistemas de segurança da informação devem garantir que elas sejam verificáveis, completas, úteis e eficazes (LAUDON, 2003).

Quando se fala de troca de informações surgem algumas questões e uma delas diz respeito aos domínios em que estas informações estão. Isso é importante porque um cliente que está em determinado domínio e tenta trocar informação com outro cliente e/ou que esteja noutro domínio pode incorrer no problema de *cross domain*. Geralmente os departamentos de TI das empresas – e também os grandes fabricantes de *software* e *hardware* – estabelecem em suas políticas de privacidade que isso não seja permitido, salvo se haver sido previamente acordado entre as partes.

O método de desenvolvimento apresentado nesta pesquisa deriva da leitura cuidadosa das normas ISO 27.001 (ABNT, 2006) e ISO 17.799 (ABNT, 2005), da escolha de um problema específico que afete a segurança de empresas altamente dependentes de software e da criação e utilização de uma solução que resolva o referido problema; o que

neste caso, foi a necessidade de sistemas complexos utilizarem múltiplos domínios, ou *cross domain*. Ao longo do texto, será mostrado que é possível detectar e resolver os problemas relacionados a *cross domain* nos *browsers*.

Este problema pode acontecer até mesmo localmente, pois empresas altamente dependentes de *software* e/ou que utilizem aplicações complexas podem possuir sistemas que são executados em domínios diferentes, e que, por alguma razão, pode haver necessidade de haver chamadas e trocas de informação entre elas. Neste caso, mesmo que a corporação seja a mesma, o problema de *cross domain* acontece e precisa ser tratado.

Para lidar com esta situação, existem soluções *cross domain*, que são, basicamente, mecanismos computacionais que aumentam a capacidade de acesso e transferência de informações entre dois ou mais domínios de forma segura; de forma manual ou automatizada. Os sistemas podem utilizar, inclusive, software de terceiros ou hardware integrados em domínios de segurança completamente incompatíveis e mesmo com níveis de classificação diferentes. A Figura 1 ilustra uma situação em que o problema de *cross domain* acontece.

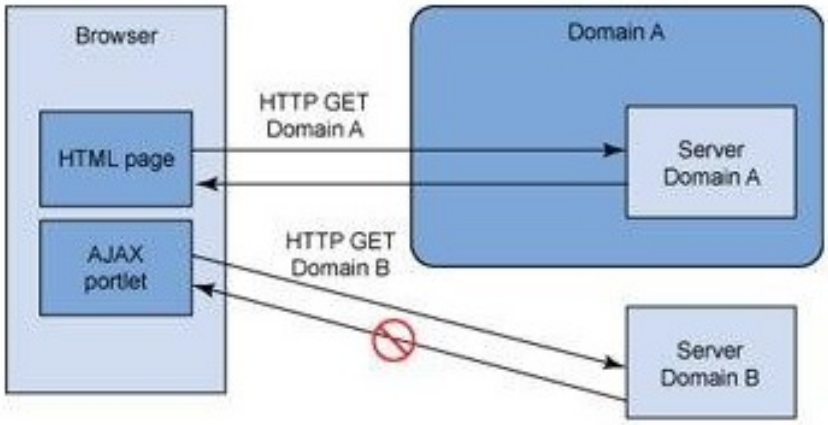

Figura 1 – Requisições HTTP GET realizadas *cross domain*.

Na situação representada, duas requisições HTTP GET são feitas pelo *browser*. A primeira tem origem no HTML da página e se destina a um servidor que está no domínio A. Já a segunda, se origina em um *portlet* AJAX e visa um servidor no domínio B.

1.1 Hipótese

É possível detectar e resolver problemas relacionados com a segurança da informação em aplicações cross-domain.

1.2 Objetivos Gerais

O objetivo principal deste trabalho é demonstrar como é possível detectar e resolver de algumas meneiras os problemas relacionados com cross-domain nos browsers e assim escolher quais desses métodos podem ser considerados melhores. São propostos como objetivos específicos :

- Detectar os problemas de cross-domain.
- Encontrar as soluções para o problema.
- Escolher as melhores soluções.

1.3 Visão Geral do Trabalho

O desenvolvimento da pesquisa que resultou na criação de uma solução para os problemas de segurança da informação relacionados ao *cross domain* foi dividida em três grandes fases:

1. Na primeira fase da pesquisa, foi realizado extensivo levantamento da literatura sobre os fundamentos e conceitos básicos de segurança da informação, vulnerabilidade e tipos de ataque.
2. Na segunda fase, foram estudadas ferramentas de *hardware* e *software* utilizadas no mercado para resolver alguns problemas de segurança, inclusive o de *cross domain*.
3. Por fim, durante a terceira fase, foi desenvolvida uma solução para o problema de *cross domain* e a criação de uma aplicação piloto para validá-la.

Esta monografia está dividida em seções sendo que a primeira seção do documento apresenta a fundamentação teórica sobre segurança da informação, vulnerabilidade e tipos de ataque aos sistemas computacionais. Já a terceira seção é responsável por documentar alguns trabalhos relacionados à pesquisa. Por sua vez, a quarta seção detalha os métodos de resolução de *cross domain* e apresenta a solução criada durante o desenvolvimento da pesquisa. Por fim, a seção de considerações finais apresenta os resultados obtidos, desafios encontrados e possibilidades de trabalhos futuros.

2 Trabalhos Relacionados

As pesquisas relacionadas a segurança da informação em aplicações cross-domain foram iniciadas mais recentemente, quando perceberam a grande necessidade de se tratar essa questão quando trabalhamos com dominío cruzado. Temos alguns trabalhos relacionados com soluções em aplicações cross-domain que serão citados a seguir.

Recentemente foi criado o SecureOne que foi proposto por Rockwell Collins (COLLINS, 2011), onde é tratada desta questão de domínio cruzado em sistemas táticos militares.

Os autores Nikolas Selimis, Joshua Testerman, Jamie Lafleur-Vetter e Thomas Ruoff (SELIMIS JOSHUA TESTERMAN, 2007) propuseram sistemas e métodos para fornecer um cross-domain, capacidade de navegação segura usando um terminal remoto. A Solução Cross-Domain (CDS) permite que os usuários de um domínio possam navegar um material(fazer transferências) operações com os mesmos privilégios que qualquer outro usuário de outro domínio. Essa transferência de arquivos é assegurada pela (AFT) transferência de arquivos Assegurada, onde deve-se assumir que os arquivos contém somente as informações devidamente classificadas. AFT prevê um canal bi-direcional para essa troca de arquivos.

O trabalho de Jun Yang, Rong Yan e Alexander G. Hauptmann (YANG RONG YAN, 2007) tem foco em aplicações multimídia, onde ele usa técnicas de adaptação para dados com distribuições diferentes. O cross-domain de vídeo de detecção, visa adaptar o conceito de classificadores em vários domínios de vídeo. O método proposto pelo autor visa superar várias linha de base e métodos concorrentes em termos de precisão da classificação e eficiência em cross-domain voltado para multimídia.

Collin Jackson e Helen J. Wang (JACKSON, 2007) foca na combinação de dados e código de fontes de terceiros que permite uma nova onda de mashups da web que agregam criatividade e funcionalidade para aplicações web. No entanto, os navegadores são mal concebidos para transmitir dados entre domínios, muitas vezes obrigando os desenvolvedores web a abandonar a segurança em nome da funcionalidade. Para melhorar esta deficiência, desenvolveram Subspace, um mecanismo de comunicação entre domínios que permite a comunicação eficiente entre domínios sem sacrificar a segurança. O protótipo dos autores requer apenas uma biblioteca JavaScript pequena, e funciona em todos os principais navegadores. Eles acreditam que Subspace pode servir como uma nova comunicação segura primitivo para mashups da web.

No trabalho de Karl MacMillan, Spencer Shimko, Chad Sellers, Frank Mayer e Art Wilson (MACMILLAN SPENCER SHIMKO; WILSON, 2006) tem-se a construção de sistemas de computador que permitem a transferência controlada de dados entre domínios de segurança, comumente chamados soluções cross-domain (CDS) ou guardas, onde é muito comum apresentar desafios exclusivos de segurança. No artigo, os autores exploraram as lições aprendidas a partir de uma construção de sistemas de vários CDS no SELinux. Exploraram as propriedades de segurança, definiram o papel do sistema operacional para implementar essas propriedades de segurança, e descreveram a suas experiências com o uso do SELinux para cumprir a função de sistema operacional.

Anil Saraswathy e Steve Tillery (SARASWATHY, 2007) trabalharam com o método de provisionamento de domínio, onde se tem um sistema e uma arquiterua para gerenciar com segurança a ampla variedade de sistemas de TI, fornecendo administração unificada, implementação e auditoria, e conectividade simplificada. O uso combinado de certos aspectos do ilustrativo IDM Provisioning Platform (DataForumTM), Conectividade Component Architecture, em tempo de criação de uma ferramenta de fluxo de trabalho do cliente, bem como a utilização de certificados digitais para garantir canais de comunicação entre domínios, onde coletivamente oferecem uma abordagem única para a solução de provisionamento de problema em domínio cruzado.

Hsiao, Shun-Wen and Sun, Yeali S. and Ao, Fu-Chi and Chen e Meng Chang (HSIAO et al., 2011) tem um trabalho voltado para segurança em mashup web, que essencialmente é uma aplicação web que integra o conteúdo de fontes heterogêneas para fornecer aos usuários uma experiência mais integrada e transparente de navegação. Mashups do lado do cliente diferem de mashups do lado do servidor em que o conteúdo é integrado no navegador, usando os scripts do lado do cliente. No entanto, o legado da política de mesma origem (SOP) implementado pelos navegadores não podem fornecer um mecanismo de comunicação do lado do cliente flexível para a troca de informações entre diferentes fontes, daí surge as soluções cross-domain(CDS). Para resolver este problema, foi proposto um modelo de comunicação de domínio cruzado do lado do cliente facilitado por um proxy confiável e o postMessage método de mensagem do HTML5. O design baseado em proxy permite aos usuários navegar mashups sem instalar plug-ins. Para mashups promotores, o API fornecida minimiza a quantidade de modificação do código. Os resultados das experiências demonstraram que a sobrecarga gerada pelo modelo baseado em proxy é baixa e razoável.

3 Fundamentação Teórica

Este capítulo apresenta os fundamentos e conceitos básicos de segurança da informação; as ferramentas de *hardware* e *software* relevantes ao entendimento dos diversos aspectos envolvidos na pesquisa e também as técnicas empregadas em projetos de segurança.

A segurança da informação é um tema essencial para qualquer empresa já que a sociedade moderna vive um momento de utilização da informação maior que em qualquer outra época da civilização humana. Isso exige que se tenha uma infraestrutura de comunicação disponível que suporte o volume de todas as transações executadas e que também permita o armazenamento dos dados de forma segura.

O valor da informação corporativa não é facilmente mensurável devido à quantidade crescente de dados de diversas fontes que as empresas possuem. Por isso, é essencial identificar todos os elementos que compõem as necessidades de comunicação de dados (LEE, 2000), os quais podem ser divididos nas seguintes categorias de ativos:

- Informações: dados armazenados em meio magnético ou físico, como relatórios, planilhas e arquivos de configurações.
- Infraestrutura de suporte: computadores, mídias, elementos de rede e softwares de computador.
- Pessoas: todos os indivíduos que manipulam e/ou utilizam as informações.
- Estruturas organizacionais: imóveis, mesas, armários, etc.

O capítulo é dividido em duas seções. A primeira apresenta os fundamentos e conceitos teóricos de segurança da informação e conta com tópicos como princípios, ameaças, vulnerabilidades, riscos, atacantes, tipos de ataques e ferramentas utilizadas em segurança. A segunda seção, por sua vez, é dedicada aos conceitos técnicos, ferramentas e outras tecnologias referentes ao problema específico de *cross domain*.

3.1 Fundamentos Teóricos

Esta seção descreve os fundamentos e conceitos teóricos essenciais à compreensão do trabalho, dentre eles: princípios, ameaças, vulnerabilidades, riscos, atacantes, tipos de ataques e ferramentas utilizadas em segurança.

3.1.1 Princípios da Segurança da Informação

Os ativos de informação devem ser protegidos contra ameaças para garantir os três principios básicos da segurança da informação que são:

- Confidencialidade: as informações devem ser conhecidas apenas por aqueles que possuam as permissões de acesso necessárias; o que dificulta o "vazamento" de informações e práticas de espionagem industrial.
- Integridade: as informações devem ser mantidas no seu estado original, sem alterações, garantindo a quem as receber, a certeza de que não foram falsificadas, corrompidas ou alteradas.
- Disponibilidade: o acesso a todos os dados no momento em que sua utilização for necessária.

3.1.2 Ameaças

Ameaças são elementos que tem condição de explorar vulnerabilidades e causar problemas severos aos ativos de uma empresa (MODULO, 2007). Os ativos estão continuamente expostos a ameaças existentes, que podem colocar em risco os três princípios da segurança. Algumas categorias de ameaças são:

- Naturais: condições da natureza que podem causar danos como, por exemplo: incêndio, enchentes, terremotos, etc.
- Intencionais: decorrentes de ações propositais como disseminação de vírus, espionagem, fraude, vandalismo, roubo, etc.
- Involuntárias: originadas por falhas não intencionais dos usuários como acidentes, erros, falta de conhecimento, etc.

3.1.3 Vulnerabilidades

Vulnerabilidades são deficiências nos mecanismos de proteção que podem ter diversas origens. Elas nem sempre são identificadas ou tratadas devidamente a tempo de evitar um ataque. Algumas das fontes de vulnerabilidades são apresentadas na lista a seguir.

- Agentes da natureza como umidade, poeira, poluição e calor; e também, fatores geográficos que possam resultar em ameaças tais como instalações próximas a rios que podem causar inundações.

- Hardwares: falhas no dimensionamento do equipamento a ser utilizado, problemas de projeto e manutenção.
- Softwares: falhas no desenvolvimento que permitem a inclusão e execução de softwares com código malicioso.
- Meios de comunicação: problemas no cabeamento, antenas de radio inadequadas entre outros problemas na infraestrutura de comunicação.
- Humanas: danos que as pessoas podem causar devido a espionagem, má utilização, falta de treinamento, insatisfação com o trabalho, erros, etc.

3.1.4 Riscos

Riscos se referem à possibilidade das ameaças explorarem as vulnerabilidades, ocasionando danos ou perdas de dados, gerando prejuízos aos negócios da empresa e que acabam por afetar os princípios de confidencialidade, integridade e disponibilidade. Existem formas de se analisar os riscos, sendo que estudos que classifiquem as informações em categorias permite avaliar o impacto que uma ameaça pode gerar.

3.1.5 Ataques

Os ataques podem ocorrer quando as vulnerabilidades não são tratadas de modo contornar ou mesmo corrigir os problemas conhecidos. Coletar informações da ocorrência dos tipos de ataques é um passo necessário para dar início a um piano de ação de segurança da informação. A seguir é apresentada uma lista de ataques bem comuns:

- Ataque físico: caracterizado pelo roubo de equipamentos discos, fitas magnéticas, CD-ROM, disquetes ou outros meios de armazenamento de dados que são retirados da empresa para posterior analise ou destruição.
- Ataque a nível da aplicação: explore as vulnerabilidades dos *softwares* que utilizam o protocolo TCP-IP.
- Ataque a servidor web: caracterizado pela exploração de vulnerabilidades em softwares para publicação de páginas *web* como o IIS.
- *Buffer Overflow*: falha de controle da área de armazenamento temporário em memória RAM (*buffer*), também conhecidos como estouro de pilha e causam faihas em aplicações deixado-as indisponíveis.
- *Exploit*: software de código malicioso que explora vulnerabilidades dos mais diversos softwares como Internte Explorer e Firefox. Caracteriza-se por causar mau funcionamento, ou mesmo a indisponibilidade da aplicação.

- SQL-Injection: é uma técnica que possibilita a inserção de um código na linguagem SQL em páginas web, permitindo ao invasor o acesso aos servidores.
- *Denial of Service* (DoS): ataque de negação de serviço, responsável por sobrecarregar servidores com grande volume de informação, causando a parada do sistema operacional e sobrecarga de operações do processador.
- Packet Sniffing: software que executa a captura de pacotes IP que podem conter informações importantes de softwares de bate-papo ou mesmo softwares de e-mail.
- Scan: Também conhecido como Port Scanning, analisa portas IP que possuem servigos associados, como por exemplo, telnet.

3.2 Fundamentos Técnicos

Esta seção descreve os fundamentos técnicos essenciais como por exemplo Ajax, JQuery, crossdomais.xml, Jquery PostMessage, JSonP, Same Origin Policy.

3.2.1 Ajax

O significado de Ajax é "Asynchronous JavaScript and XML", e ele foi criado por Jesse James Garrett, em seu artigo : Uma Nova Abordagem para aplicações web (SHANNON, 2014). Pode-se dizer que AJAX essencialmente é fazer com que uma página seja carregada e renderizada com os recursos de scripts rodando no lado do cliente e carregando dados em background sem a necessidade de recarregar a página. Ajax se resume nos seguintes itens abaixo.

- Assíncrono : Fazer a comunicação e transmissão de dados sem a presença da linha de clock(ou sinal de relógio externo). Isto significa que quando você envia um request, você espera ele voltar, mas pode executar outras ações enquanto espera, sendo assim o processo não fica bloqueado. Na maioria das vezes que solicitar um request a resposta não vai voltar imediatamente, de modo que deve ter uma função configurada que irá aguardar a resposta a ser enviada de volta ao cliente, entender o seu retorno e realizar as ações configuradas. Abaixo temos a figura 2 que mostra como funciona a comunicação assíncrona.

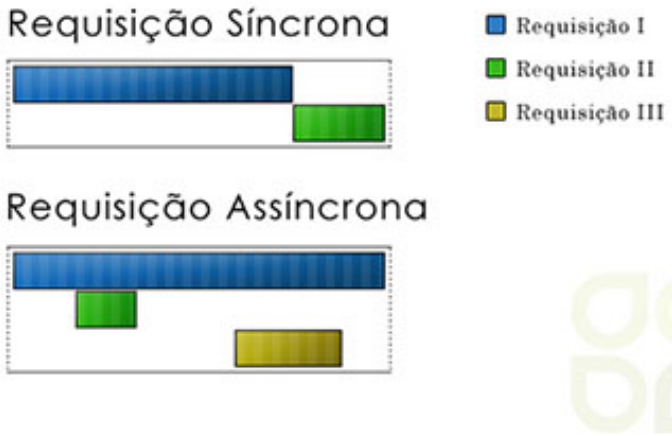

Figura 2 – Comparação comunicação Assíncrona e Síncrona.

- JavaScript : São os scripts que podem ser executados do lado do cliente e interagir com o usuário sem a necessidade deste script passar pelo servidor. É considerada a principal linguagem para programação client-side em navegadores web. JavaScript pode ser usado para modificar modelo de objeto de documento da página atual de alguma forma para mostrar ao usuário que a apresentação passou com sucesso.
- XML : Os dados que recebe de volta a partir do servidor, muitas vezes, ser empacotado como um trecho de XML, de modo que ele pode ser facilmente processado com JavaScript. Estes dados podem ser o que quiser, e contanto que você quiser.

3.2.2 JQuery

JQuery é um framework (biblioteca) de Javascript. Foi desenvolvido para simplificar e facilitar os programadores com os scripts client-side que interagem com o HTML. Ele foi lançado em dezembro de 2006 em Nova York por John Resig. Usada por cerca de 77 por cento dos 10 mil sites mais visitados do mundo, jQuery é a mais popular das bibliotecas JavaScript. Pode ser usado para fazer praticamente qualquer efeito javaScript, ou requisições em AJAX ou até mudanças na página após o seu carregamento.

3.2.3 CROSSDOMAIN.XML

O flash player bloqueia o acesso a dados entre domínios diferentes, isso para garantir que seu conteúdo não seja acessado por terceiros sem permissão. Esse erro pode ser até disparado dentro do próprio site,por exemplo, você tentar carregar uma imagem dentro do seu site pelo seu caminho completo http://www.meusite.com.br/minhaImagem.jpg e

ao carregar o site no endereço http://meusite.com.br/ sem o www. ele considera que o conteúdo está sendo acessado de dominios diferentes. Nesse caso é necessário informar ao flash player que é previsto esse acesso ao conteúdo de outro domínio. Esses dados nos podemos passar por um arquivo xml que fica na raiz de nosso site, o crossdomain.xml.

3.2.4 Jquery PostMessage

jQuery postMessage permite a comunicação simples e fácil entre diferentes dominíos utilizando window.postMessage em navegadores que suportam (FF3, Safari 4, o IE8), ao cair de volta para um método de comunicação document.location.hash para todos os outros navegadores (IE6, IE7, Opera). Com a adição do window.postMessage método, JavaScript, finalmente, tem um meio fantástico para estrutura de comunicação de domínio cruzado (ALMAN, 2014). Infelizmente, este método não é suportado em todos os navegadores. Um exemplo onde este plugin é útil é quando um Iframe filho precisa passar algum conteúdo ao seu iframe pai ou pedir a ele pra envocar alguma ação. Na figura 3 temos um exemplo de página iframe alterando documento pai em outro domínio e/ou porta.

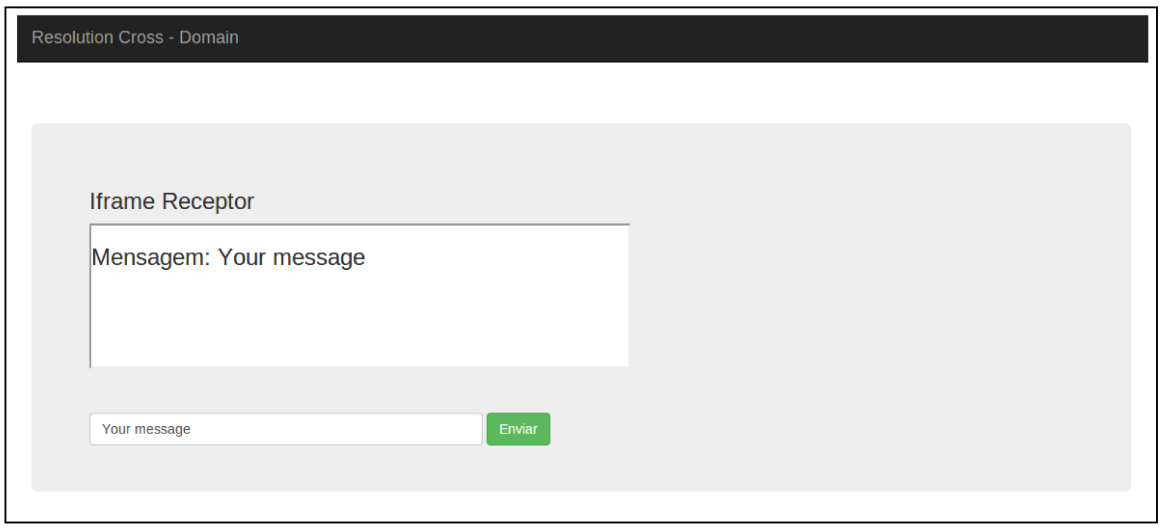

Figura 3 – Chamada via Jquery postMessage.

3.2.5 JSonP

JSONP ou "JSON with padding"é um complemento ao formato de dados JSON, a figura 4 mostra melhor como ele é representado. Ele provê um método para enviar requisições de dados de um servidor para um domínio diferente, uma coisa proibida pelos navegadores típicos por causa da *Same Origin Policy*.

Com a Política de mesma origem, uma página com o domínio dominio.example.com não pode normalmente se conectar ou se comunicar com servidores diferentes de dominio.example.com Uma exceção é a tag HTML <script>. Explorando a política aberta para a tag <script>, algumas páginas usam a tag para receber código Javascript que opera com dados em formato JSON de outra origem gerados dinamicamente. Esse padrão de uso é conhecido com JSONP. Requisições de JSONP não trazem JSON, mas código Javascript arbitrário. Eles são executados pelo interpretador Javascript, e não parseados pelo parser JSON.

Figura 4 – Representação do formato de dados usando JSONP.

3.2.6 HTTP

O HTTP é um protocolo de comunicação (na camada de aplicação segundo o Modelo OSI) utilizado para sistemas de informação de hipermídia, distribuídos e colaborativos.É a base para a comunicação de dados da World Wide Web. Hipertexto é o texto estruturado que utiliza ligações lógicas (hiperlinks) entre nós contendo texto. O HTTP é o protocolo para a troca ou transferência de hipertexto essencialmente em formato HTML entre um navegador (cliente) e um servidor web. Para que a comunicação ocorra temos os métodos GET, POST, HEAD, PUT, DELETE, TRACE, OPTIONS e CONNECT. Abaixo será detalhado os metódos principais POST e GET.

- GET: O método GET utiliza a própria URI (normalmente chamada de URL) para enviar dados ao servidor, quando enviamos um formulário pelo método GET, o navegador pega as informações do formulário e coloca junto com a URI de onde o formulário vai ser enviado e envia, separando o endereço da URI dos dados do formulário por um "?" (ponto de interrogação).Quando você busca algo no Google,

ele faz uma requisição utilizando o método GET, você pode ver na barra de endereço do seu navegador que o endereço ficou com um ponto de interrogação no meio, e depois do ponto de interrogação você pode ler, dentre outros caracteres, o que você pesquisou no Google.

- POST : O método POST envia os dados colocando-os no corpo da mensagem. Ele deixa a URI separada dos dados que serão enviados e com isso podemos enviar qualquer tipo de dados por esse método. Quando você faz um registro em um formulário e depois de enviar a URI não tem o ponto de interrogação separando os dados que você digitou, provavelmente o formulário foi enviado pelo método POST.

  Como foi dito pode-se notar que os métodos POST e GET se diferem em alguns pontos importantes, onde resumidamente a diferenças dos dois métodos é que o método GET utiliza a URL para enviar dados ao servidor e o POST envia os dados colocando-os no corpos da mensagem. A figura abaixo exemplifica a diferença dos métodos.

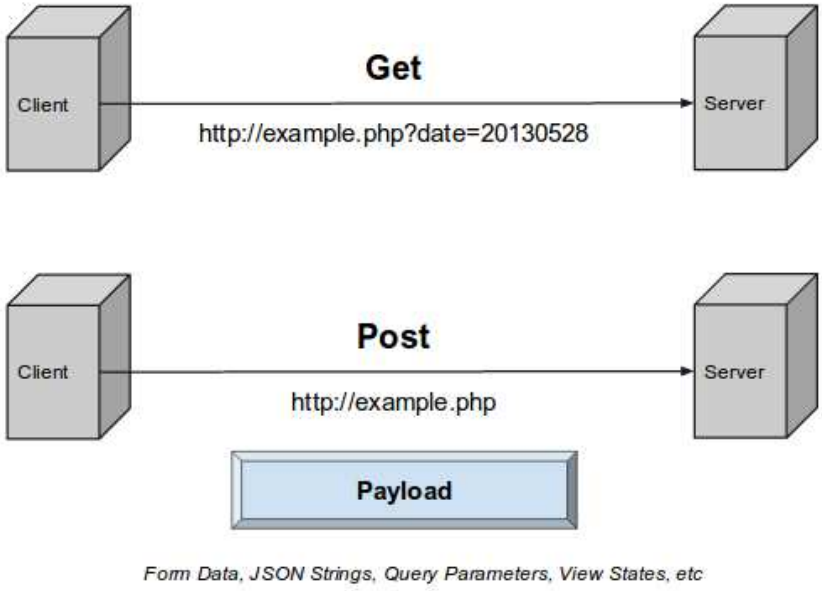

Figura 5 – Diferença entre POST e GET.

3.2.7 *Same Origin Policy*

A política de mesma origem é um conceito importante no modelo de segurança das aplicações web. Essa política permite a execução de scripts em páginas originadas no mesmo local com um esquema de combinação com o nome de host e número de porta para acessar um do outro DOM, sem restrições específicas, mas impede o acesso ao DOM. Em locais diferentes a política de mesma origem também se aplica a XMLHttpRequest e WebSocket.

Este mecanismo tem um significado especial para aplicações web modernas, que dependem amplamente cookies HTTP para manter sessões de usuário autenticado, como servidores de agir com base na informação de cookie HTTP para revelar informações sensíveis ou executar ações de mudança de estado. A estrita separação entre conteúdo fornecido por sites independentes deve ser mantida no lado do cliente para evitar a perda de confidencialidade de dados ou integridade.

A política de mesma origem permite solicitações HTTP inter-origem com os métodos GET e POST, mas nega inter-origem PUT e DELETE solicitações. Além disso, a origem pode usar cabeçalhos HTTP personalizados ao enviar pedidos de si, mas não pode usar cabeçalhos personalizados ao enviar pedidos de outras origens.

Duas páginas têm a mesma origem, se o protocolo, a porta (se for especificado), e inúmeras são as mesmas para ambas as páginas. A tabela a seguir apresenta exemplos de comparações de origem para a URL http://store.company.com/dir/page.html.

URL	Resultado	Motivo
http://store.company.com/dir2/other.html	Sucesso	*
http://store.company.com/dir/inner/another.html	Sucesso	*
https://store.company.com/secure.html	Falha	Protocolo diferente
http://store.company.com:81/dir/etc.html	Falha	porta diferente
http://news.company.com/dir/other.html	Falha	host diferente

Tabela 1 – Tabela de comparação da mesma origem.

4 Desenvolvimento e Método

Um dos maiores problemas quando se trata de dominío cruzado são os iframes inseridos em páginas onde pode surgir uma necessidade de haver uma comunicação entre o parent(aplicação principal em que o iframe está inserido) e o iframe. Pode ser por exemplo, a situação em que o iframe deve ser fechado quando o usuário executar uma ação, note que está ação está sendo feita dentro do iframe que está em outro domínio, parte deste presuporto estamos tendo um problema de cruzamento de domínio. A figura abaixo 6 mostra o cenário descrito acima.

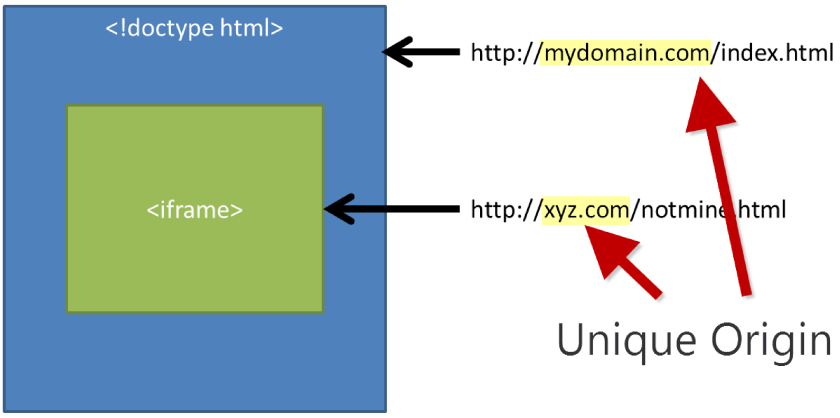

Figura 6 – Parent e Iframe.

Uma solução para este problema seria usar scripts modernos que são capazes de fazer essa comunicação de forma transparente sem que usuário percebe, como se tudo fosse apenas uma aplicação e não uma ou mais aplicações integradas. Na figura 7 tem-se a demonstração de como isto é feito.

Além disso, quando há necessidade de trabalhar com cross domain usando transferências de mensagens, arquivos e informações deve-se ter em mente a necessidade de se fazer isso de forma segura. Nos browsers de hoje em dia tem-se implementado Same Origin Policy, onde se tem uma segurança maior no sentido de que para fazer uma alteração, ou manipulação de uma aplicação que está em outro domínio deve-se usar de algumas tecnicas que serão apresentadas a seguir.

- PostMessage : Usando este método os domínios se comunicam por meio de uma troca de mensagens como exemplificado a seguir.

```
win.postMessage("mensagem aqui","host:port" );
```

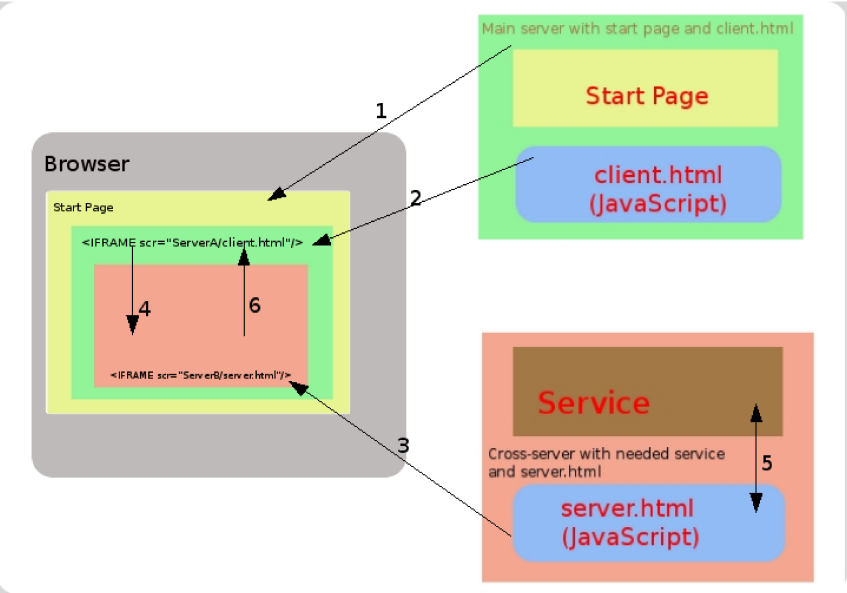

Figura 7 – Scripts.

No lado do servidor temos o código abaixo.

```
<script>
function displayMensage(event){
   if(event.origin !== 'http://scriptandstyle.com:8080') {
   console.log("Error : Different host (exact match required) ");
   return;
   }
   document.getElementById("msg").innerHTML = "Mensagem: " + event.data;
}

if (window.addEventListener){
  addEventListener("message", displayMensage, false)
} else {
  attachEvent("onmessage", displayMensage)
}
</script>
```

Note que o servidor tem um addEventListener que fica esperando a chegada de mensagem enviadas pelo cliente, quando essa mensagem chega ele valida de qual host a mesma esta vindo, caso seja um host esperado ele faz o foi definido para ser realizado.

- XML : Usando o arquivo crossdomain.xml o servidor dá as permisssões dos domínios que podem fazer alterações, ou seja, como se fosse um acordo entre as partes envolvidas. Abaixo tem-se um exemplo de uso do arquivo crossdomain.xml.

```
<?xml version="1.0"?>
<cross-domain-policy>
    <allow-access-from domain="*.meusite.com" />
    <allow-access-from domain="www.siteparceiro.com" />
    <allow-access-from domain="192.0.34.122" />
</cross-domain-policy>
```

  Na tag allow-access-from colocamos o domínio que poderá acessar os dados do servidor mesmo estando em outro domínio, como pode-se ver esse arquivo só pode ser editado pelas pessoas com permissão.

- Ajax e JSONP: O ajax por questões de segurança, não é cross-domain, ou seja, não permite que um site interaja com um script de outro site, pois quebra algumas regras de segurança e abriria brechas para possíveis invasões. Mas tem como fazer essa interação usando JSONP como é demonstrado no exemplo a seguir.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta http-equiv="content-language" content="pt-br" />
<script src="jquery.js"></script>
<script src="jquery.xdomainajax.js"></script>
<script type="text/javascript">
$.ajax({
    url: 'http://www.lucaspeperaio.com.br',
    type: 'GET',
    success: function(res) {
    var headline = $(res.responseText).text();
    document.write(headline);
    }
});
</script>
</head>
<body>
</body>
</html>
```

- Usando um servidor como PROXY : A ideía de se usar um servidor como proxy para fazer o cruzamento de domínios se torna muito interessante em função da segurança que o mesmo proporciona, onde os domínios não se interagem diretamente. A figura abaixo 8 mostra como é feito usando está solução.

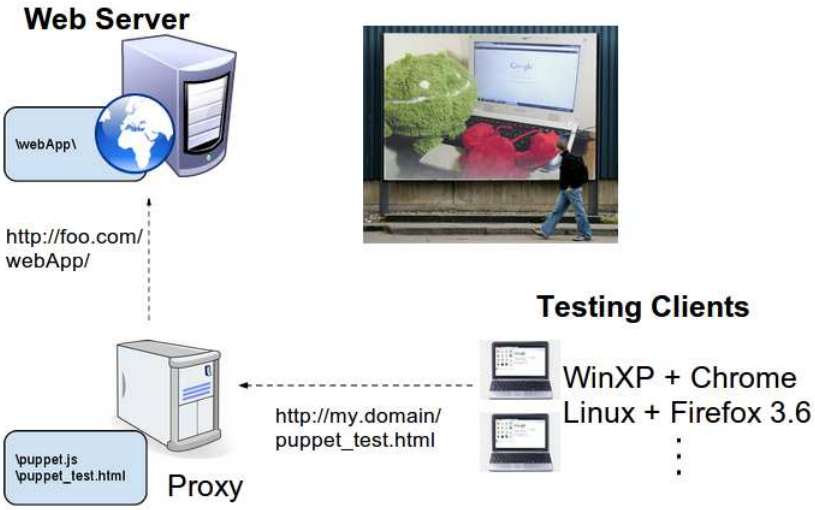

Figura 8 – Solução usando proxy.

4.1 Vídeos

Quanto tratamos de vídeos esse problema também se faz presente. Em vídeos a situação é um pouco diferente, na figura 9 mostramos como é feita a transferência de vídeos em diferentes domínios. O problema surge quando um domínio tem que tranferir um vídeo pra outro domínio.

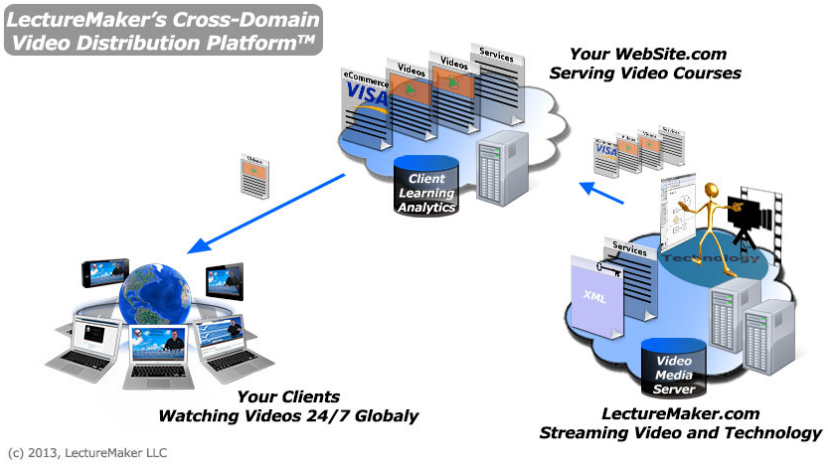

Figura 9 – Scripts.

4.2 Proxy

Essa solução pode ser considerada a mais moderna quando se trata de cross-domain, ela além de ser segura dificultando os ataques, faz com que a comunicação seja mais transparente ainda. Utilizamos XMLHttpRequest no servidor proxy e quando queremos fazer algo em outro dominío e vice-versa, passamos por ele, mapeamos a ação e os domínios. A figura faz a representação do proxy entre os domínios.

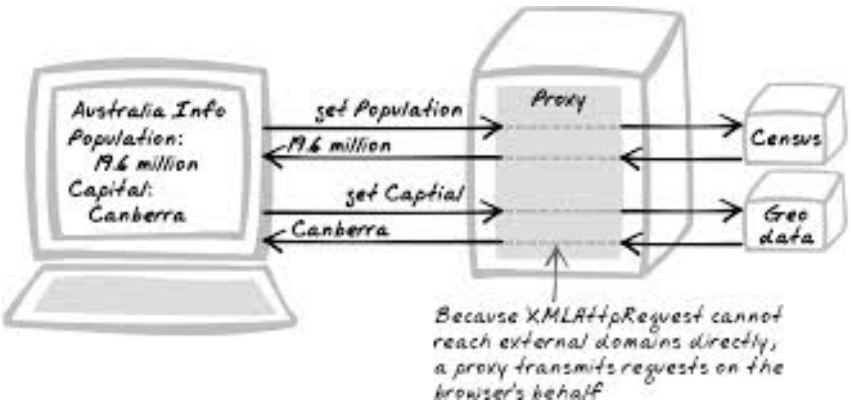

Figura 10 – Utilização de Proxy.

4.3 JSON with Padding - JSONP

Essa técnica é mais uma "solução alternativa"do que uma solução propriamente dita - mas é popular o bastante para ser suportada nas principais frameworks web (como o jQuery), e antes do CORS era uma das únicas maneiras de fazer isso sem recorrer a plugins externos ou funcionalidades específicas de um navegador. É simples de implementar, e funciona em qualquer navegador, embora não seja propriamente uma solução "segura"(só deve ser usado se o site para onde se faz a requisição é confiável). O JSONP se aproveita do fato que algumas tag são isentas da política de mesma origem (Same-Origin Policy) - dentre elas a script. Como o JSON é um formato popular de transferência de dados, e o mesmo é um subconjunto dos literais de objetos de JavaScript, poderia-se transmitir os dados desse tipo:

```
{"Name": "Foo", "Id" : 1234, "Rank": 7}
```

Usando este formato:

```
minhaFuncao({"Name": "Foo", "Id" : 1234, "Rank": 7});
```

Para isso, basta criar uma tag script especificando qual função você quer que seja chamada (o equivalente ao callback do Ajax) e colocar a requisição no atributo src. Exemplo (na prática, use o formato suportado pelo seu servidor):

```
var script = document.createElement('script');
script.src = 'http://example.com/jsonp?callback=minhaFuncao';
// + outros parâmetros
document.getElementsByTagName('head')[0].appendChild(script);
```

E no servidor, basta serializar sua resposta como JSON e depois "envolvê-la"no callback especificado. Exemplo (Django):

```
objeto = {"Name": "Foo", "Id" : 1234, "Rank": 7}
codificado_json = json.dumps(objeto)
callback = request.GET["callback"]
# ex.: "minhaFuncao"
resposta = callback + "(" + codificado_json + ");"
# "minhaFuncao({...})
return HttpResponse(resposta, mimetype="text/javascript")
```

4.4 Cookie

Os cookies são muito utilizados na web, e podem ser utilizados inclusive para fazer SSO (Single sign-on). Um exemplo disto é quando temos duas aplicação e quando o usuário logar em uma, ele automaticamente estará logado na outra. Na figura 11 dois sistemas tentam compartilhar cookies para fazer o SSO.

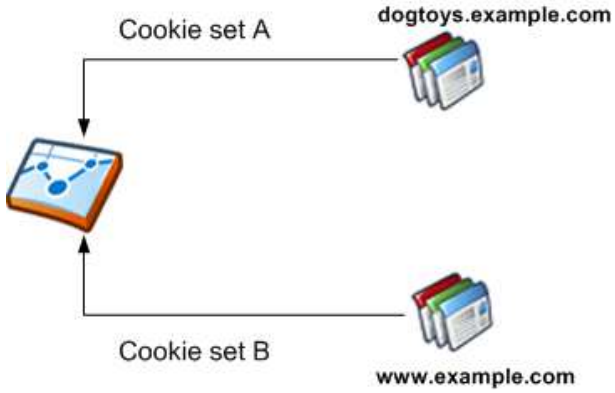

Figura 11 – Compartilhamento de cookies.

A transferência de cookies pode ser feita utilizando-se de alguns passos bem simples que serão descritos a seguir.

- centralizar todos Cokies em um único domínio, digamos cookiemaker.com
- quando o usuário faz uma solicitação para example.com você redirecioná-lo para cookimaker.com
- cookiemaker.com redireciona-lo de volta para example.com com a informação que você precisa

4.5 Solução Proposta usando PostMessage *versus* Proxy

Primeiramente vamos considerar a solução mais simples e rápida para resolver esse problema de cross-domain, que é usado apenas o postMessage. No postMessage como já foi dito anteriormente a comunicação é inteiramente em javascript e desta forma, basta que as duas partes tenham um minímo de conhecimento uma da outra que a comunicação ocorre. Vamos imaginar a situação em que foi desenvolvido uma aplicação a alguns anos, com algumas funcionalidades prontas e funcionais, ou seja a aplicação está estável, e foi desenvolvida uma aplicação mais recente com uma arquitetura um pouco mais moderna, e deve ser desenvolvida um funcionalidade nesta aplicação bem complexa mas que já está desenvolvida na aplicação anterior e está pronta e homologada. Desta forma precisamos de alguma forma extrair apenas a funcionalide da outra aplicação e usar. Como estamos considerando que estão partes estão de acordo com a comunicação, vamos colocar um iframe que contém a url da aplicação que tem a funcionalidade e desta forma fazer o postMessage. As subseções a seguir mostram como isso pode ser feito.

4.5.1 PostMessage

4.5.1.1 Domínio 1

Nesta parte do código considera-se apenas o elemento que corresponde ao iframe(Domínio 2) e fazer o postMessage nele. Podemos perceber que o postMessage tem o parametro da message que pode ser qualquer coisa e o domínio que esta efetuando.

```
<script>
    $(document).ready(function() {
        var win = document.getElementById("express").contentWindow;
        var domain = 'http://localhost:8080';
        $("form").submit(function() {
            win.postMessage(this.elements.msg.value,"*" );
```

```
            return false;
        });
    });
</script>
```

4.5.1.2 Domínio 2

No domínio adicionamos o EventListener que é o cara responsável por ficar escutando quando a mensagem vai ser recebida e fazer o que tem que ser feito. Neste exemplo vai apenas escreve no html a mensagem recebida do domínio 1.

```
<script>
    function displayMensage(event){
        console.log("Message received : " + event.data);
        document.getElementById("msg").innerHTML = "Mensagem: " + event.data;
    }

    if (window.addEventListener){
        addEventListener("message", displayMensage, false)
    } else {
        attachEvent("onmessage", displayMensage)
    }
</script>
```

Como pode ser visto nessas seções, as dois domínios se comunicam somente via javascript e todo o código está disponível para o cliente e isso pode se tornar um problema, se as duas aplicação forem de uma mesma empresa ou instituição as vezes essa é a solução rápida, mas se for de diferentes é conveniente fazer uma solução mais segura.

Nas seções subsequentes vai ser apresentada uma solução um pouco mais complexa e segura, que é usando um proxy como foi descrito no item 5.3.

4.5.2 Proxy

Agora vamos considerar está solução, que é bem mais interessante no ponto de vista da segurança da informação, pois vamos colocar um interceptador entre duas aplicações o chamado proxy. Esse cara vai estar todo implementado em back-end, ou seja os clients vão apenas fazer post e get neste servidor e ele vai ficar responsável de gerenciar quem pode comunicar com quem. Este servidor proxy vai ser implementado usando nodejs para fazer toda a parte de servidor e gerenciamento de rotas.

4.5.2.1 Aplicação 1

Está é a aplicação requester, ou seja a aplicação que quer pedir algum recurso a outra aplicação que está rodando em outro servidor. Vamos começar com mais ou menos o mesmo código da solução com postMessage, mas agora vamos fazer realmente fazer uma requisição rest via POST no servidor proxy requisitando o recurso. Neste momento está aplicação vai ficar com a promessa de ter um sucesso ou uma falha nesta requisição.

```
<script>

$(document).ready(function() {

  var win = document.getElementById("express").contentWindow;
  var domain = 'http://localhost:8080';

  $("form").submit(function() {
     var request = {};
     request.message = this.elements.msg.value;

    $.ajax({
        url: "http://localhost:3002/proxy/process",
        type: "POST",
        crossDomain: true,
        data: request,
        dataType: "json",
        success:function(result){
            console.log(JSON.stringify(result));
        },
        error:function(xhr,status,error){
            console.log(status);
        }
    });
 });
});

</script>
```

4.5.2.2 Servidor Proxy

Quando o POST da aplicação 1 chega no proxy ele monta o request que vai ser feito na aplicação 2 e aplica o Access-Control-Allow-Origin que nada mais é que dar a permissão de receber e enviar dados atrás de uma origem.

```
app.post('/proxy/process', function(req, res) {
var json = JSON.stringify(req.body);
var options = {
hostname: 'localhost',
port: 3001,
path: '/app2/process',
method: 'POST',
headers: {
'Content-Type': 'json',
'Content-Length': json.length
}
};

var req = http.request(options, function(resp) {

resp.on('data', function(chunk) {
     res.header("Access-Control-Allow-Origin", "http://localhost:3000");
res.header("Access-Control-Allow-Methods", "POST");
res.header("Access-Control-Max-Age", "1000");
console.log("Json post http://localhost:3001/app2/process :");
console.log('BODY: ' + chunk);
res.send(JSON.parse(chunk));
//json = chunk;
});
}).on('error', function(e) {
console.log("Got error: " + e.message);
});
// write data to request body
req.write(json);
req.end();
});
```

4.5.2.3 Aplicação 2

A aplicação 2 recebe o post enviado pelo proxy e responde para ele que foi recebido com sucesso. Neste momento temos toda a comunicação feita perfeitamente entre os domínios.

```
app.post('/app2/process', function(req, res) {
req.on('data', function(data) {
console.log("Received app1 : " + JSON.parse(data).message);
});
req.on('end', function() {
res.send({
message: 'Received OK'
});
})
});
```

As imagens a seguir mostram o que foi logado nas três aplicações. Primeiramente vamos mostrar o que foi logado no browser referente a aquelas promessas mostradas na sessão da aplicação 1.

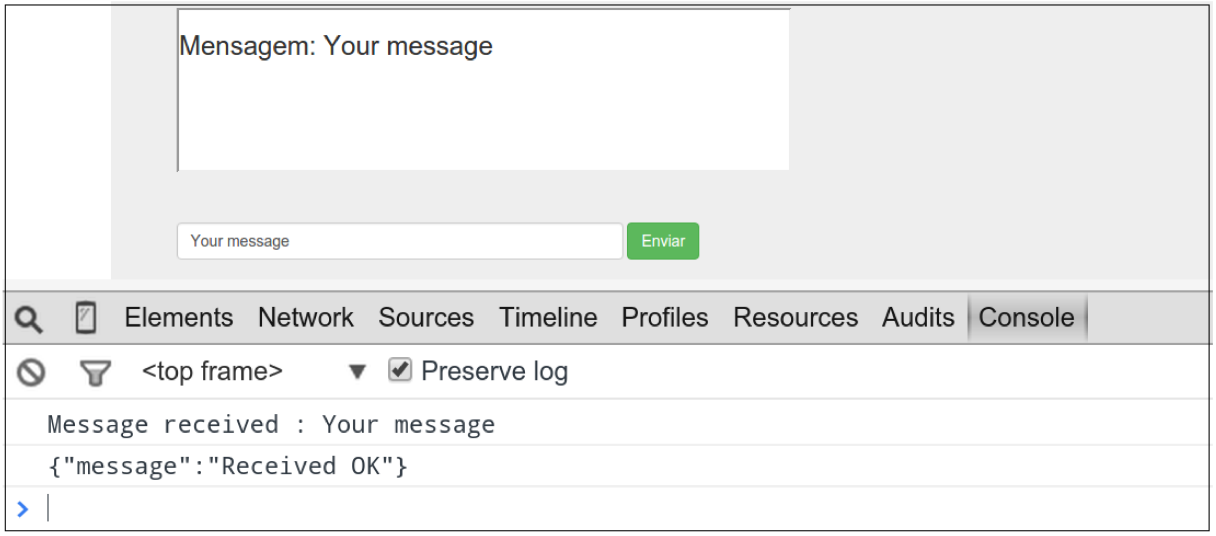

Figura 12 – Aplicação 1.

Na aplicação 2 é mostrada a mensagem enviada pela aplicação 1 passando pelo proxy.

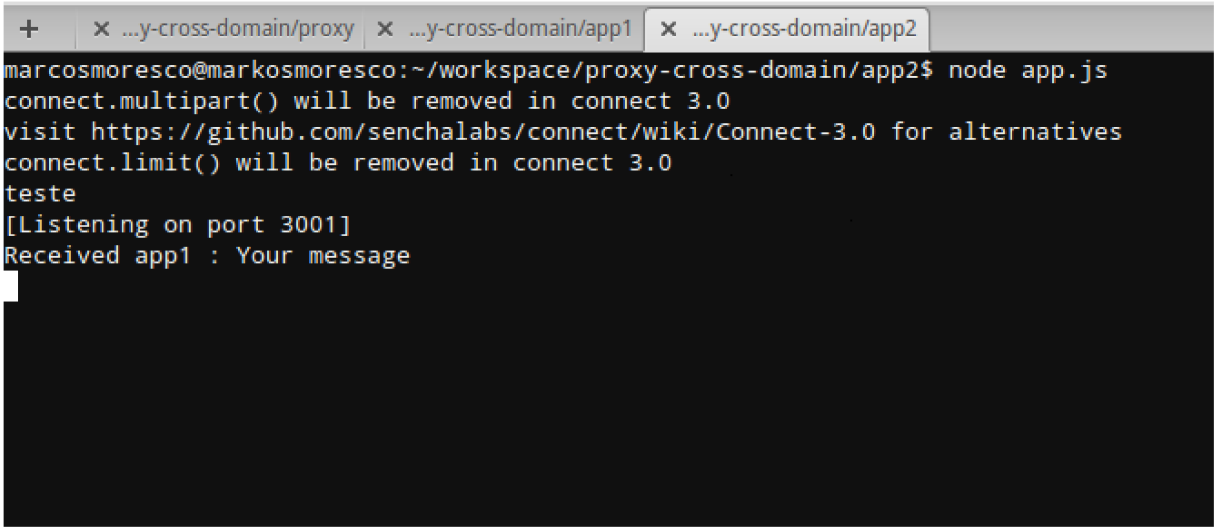

Figura 13 – Aplicação 2.

E nesta imagem mostra-se o momento em que o proxy loga a mensagem recebida e envia via POST para a aplicação 2.

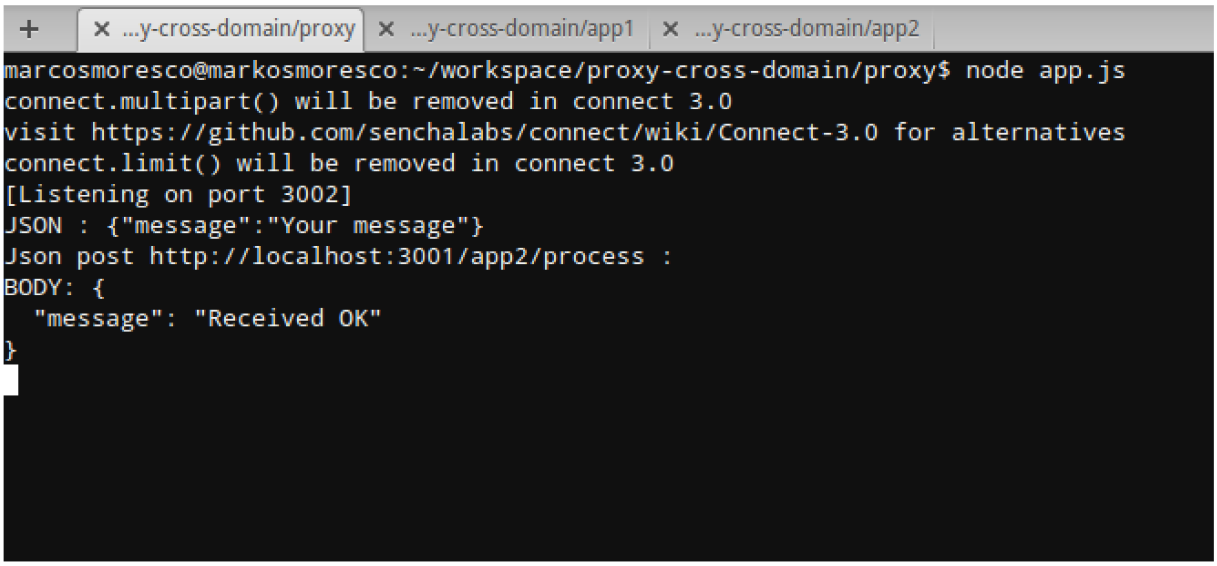

Figura 14 – Servidor Proxy.

O que pode ser concluído destas duas soluções e que o proxy esconde o código que realmente faz a comunição funcionar, isto devido ao fato dele estar todo em back-end e isso é muito importante pra quem lida com a segurança da informação. As próximas seções descrevem o processo de desenvolvimento da solução que resolve o problema de *cross domain*.

5 Considerações Finais

Neste trabalho foram descritos resultados de pesquisa que propôs uma solução para o problema de *cross domain* no contexto organizacional. Para atingir esse objetivo, inicialmente foi realizada pesquisa bibliográca que possibilitou as seguintes conclusões:

- Existem variados trabalhos e métodos de resolução de problemas de segurança, e existem também outras abordagens para a resolução de *cross domain* propostos para diversos contextos organizacionais.
- Entre documentações de implementações da resolução de *cross domain* pois um mesmo conceito pode ser identicado por diferentes termos, e em algumas documentações existem conceitos com denições imprecisas.
- Algumas propostas de resolução de *cross domain* são coleções de boas práticas com poucas referências para trabalhos cientícos, e o nível de detalhamento das suas documentações é variável.

Após a pesquisa bibliográca, foram realizadas atividades para propor uma solução do problema de *cross domain* no contexto organizacional, e o emprego dessa solução em uma prova de conceito. Essas atividades possibilitaram as seguintes conclusões:

- A construção de artefatos prescritos em frameworks de arquitetura da informação pode requerer variadas ferramentas, e o esforço para a implementação da solução pode ser signicativo para quem não está habituado com a área.
- O uso de tecnologias de mercado pode melhorar a organização da implementação de *software* que faz uso de *cross domain*.

Quanto às contribuições da pesquisa, é possível destacar a descrição de conceitos acerca de segurança de informação, ferramentas e técnicas de implementação, métodos para construção de uma solução que utilize cross-domain de forma transparente para o usuário e desenvolvedores.

No que se refere às limitações da pesquisa, destacam-se:

- A solução do problema de cross domain foi desenvolvida considerando-se um contexto específico.

- Não foram construídos todos os artefatos sugeridos por métodos de desenvolvimento de sosoftware consagradas; e os artefatos construídos foram testados apenas por um grupo de potenciais usuários.

Finalmente, algumas sugestões de pesquisas futuras:

- Criar um modelo de framework a partir da solução individual para resolver o problema de cross domain em domínios não contemplados nesta pesquisa.
- Ampliar a solução proposta com o intuito de contemplar o desenvolvimento de arquiteturas.

Referências

ABNT, A. B. de N. T. *Norma ABNT NBR ISO/IEC 17799:2005 – Código de Prática para a Gestão da Segurança da Informação*. [S.l.], 2005. Citado na página 6.

ABNT, A. B. de N. T. *Norma ABNT NBR ISO/IEC 27001:2006 — Sistemas de Gestao de Seguranga da Informação – Requisitos*. [S.l.], 2006. Citado na página 6.

ALBERTIN, L. A.; MOURA, R. de. *Administração de Informática: Funções e Fatores Críticos de Sucesso*. São Paulo: Editora Atlas, 2001. Citado na página 6.

ALMAN, B. *jQuery postMessage: Cross-domain scripting goodness*. [S.l.], 2014. Disponível em: <http://benalman.com/projects/jquery-postmessage-plugin>. Acesso em: 30 jun. 2014. Citado na página 16.

COLLINS, R. Secureone cross domain technologies. *IEEE Trans*, p. 263, 2011. Citado na página 9.

CREMONESI ANTONIO TRIPODI, R. T. P. Cross-domain recommender systems. *IEEE Trans*, 2011. Nenhuma citação no texto.

HSIAO, S.-W. et al. A secure proxy-based cross-domain communication for web mashups. In: ZAVATTARO, G.; SCHREIER, U.; PAUTASSO, C. (Ed.). *ECOWS*. IEEE, 2011. p. 57–64. ISBN 978-1-4577-1532-7. Disponível em: <http://dblp.uni-trier.de/db/conf/ecows/ecows2011.html#HsiaoSAC11>. Citado na página 10.

JACKSON, H. J. W. C. Subspace: secure cross-domain communication for web mashups. *IEEE Trans*, 2007. Citado na página 9.

KRAUS, E. *Arquivo crossdomain.xml*. [S.l.], 2014. Disponível em: <http://flex.eduardokraus.com/arquivo-crossdomain>. Acesso em: 30 jun. 2014. Nenhuma citação no texto.

LAUDON, K. *Gerenciamento de Sistemas de informação*. Rio de Janeiro: Editora LTC, 2003. Citado na página 6.

LEE, S. S. W. A framework for constructing features and models for intrusion detection systems. In: *ACM Transactions on Information and System Security, vol. 3, no. 4*. [S.l.]: ACM, 2000. Citado na página 11.

MACMILLAN SPENCER SHIMKO, C. S. F. M. K.; WILSON, A. *Lessons Learned Developing Cross-Domain Solutions*. [S.l.], 2006. Disponível em: <http://www.tresys.com/innovation/papers/Lessons-Learned-in-CDS.pdf>. Acesso em: 18 jul. 2014. Citado na página 10.

SARASWATHY, S. T. A. Cross domain provisioning methodology and apparatus. *IEEE Trans*, 2007. Citado na página 10.

SELIMIS JOSHUA TESTERMAN, J. L.-V. T. R. N. Cross-domain solution (cds) collaborate-access-browse (cab) and assured file transfer (aft). *IEEE Trans*, p. 263, 2007. Citado na página 9.

SHANNON, R. *Ajax*. [S.l.], 2014. Disponível em: <http://www.yourhtmlsource.com/javascript/ajax.html>. Acesso em: 30 jun. 2014. Citado na página 14.

SWARUP, S.; RAY, S. R. *Cross-Domain Knowledge Transfer Using Structured Representations*. [S.l.], 2006. Disponível em: <http://www.aaai.org/Papers/AAAI/2006/AAAI06-081.pdf>. Acesso em: 18 jul. 2014. Nenhuma citação no texto.

YANG RONG YAN, A. G. H. J. Cross-domain video concept detection using adaptive svms. *IEEE Trans*, p. 263, 2007. Citado na página 9.