



**FACULDADE DE TECNOLOGIA DE AMERICANA**

SEGURANÇA DA INFORMAÇÃO

# A SEGURANÇA NO LOCALSTORAGE

ANDRÉ LUIZ FERREIRA
Prof. Orientador BENEDITO CRUZ

AMERICANA / SP
2011



**FACULDADE DE TECNOLOGIA DE AMERICANA**

SEGURANÇA DA INFORMAÇÃO

# A SEGURANÇA NO LOCALSTORAGE

ANDRÉ LUIZ FERREIRA
anlferreira@live.com

Monografia desenvolvida em cumprimento à exigência curricular do Curso de Segurança da Informação da FATEC-AM, sob orientação do Prof. Benedito Cruz.

Área: Segurança da Informação

AMERICANA / SP

2011

**BANCA EXAMINADORA**

Prof. Benedito Cruz (Orientador)
Prof. Antônio da Costa Barros
Prof. Marcus Lahr

Aos meus familiares e amigos.

## AGRADECIMENTOS

Aos meus pais, Genivaldo e Denise, que mais do que me proporcionaram uma boa educação e vida acadêmica, formaram os fundamentos do meu caráter e me deram as oportunidades de ser quem sou.

Agradeço também à minha irmã Jennyfer, que sempre está lá quando mais preciso dela.

Obrigado também à Pamella Santarosa, pela cumplicidade e a compreensão em todos os momentos difíceis.

Finalmente, obrigado a André Walker, que além de marcar minha infância, foi e continua sendo uma fonte de conhecimentos e inspiração para que eu seja um profissional melhor me move a buscar sabedoria.

*“Um raciocínio lógico leva você de A a B. A imaginação*

*leva você a qualquer lugar que você quiser.”*

Albert Einstein

## RESUMO

O HTML5 é a mais recente versão do Hypertext Markup Language, a ferramenta de programação utilizada para construir sites que trás diversas novas funcionalidades, aguçando os profissionais da área. Uma das suas mais fascinantes novidades é o LocalStorage, a capacidade que os servidores tem de armazenar dados nos navegadores e computadores clientes. Contudo, o LocalStorage não é perfeito e existem falhas de segurança que podem ser exploradas por invasores. Esta publicação tem por objetivo mostrar estas falhas e como elas são exploradas, visando prover melhores soluções para a segurança na internet.

**Palavras chave:** LocalStorage. HTML5. Segurança. Riscos. Vulnerabilidades. Javascript. Injeção de dados.

## ABSTRACT

The HTML5 is the latest version of the Hypertext Markup Language, the programming tool used to build web sites which brings many new functions, improving and exciting all the web developers. One of its most fascinating improvements is the LocalStorage, the ability for the web servers to store data in client computers and browsers. Yet, the LocalStorage it is not perfect and there are many security flaws that can be exploited by attackers. This publication has the purpose to show these flaws and how they are exploited, in order to provide better solutions for the web security.


**Key words**: LocalStorage. HTML5. Security. Risks. Vulnerabilites. Javascript. Data Injection.

# SUMÁRIO

# LISTA DE ILUSTRAÇÕES

# LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| HTML5 | *Hypertext Markup Language 5* |
| CSS | *Cascading Style Sheets* |
| HTML | *Hypertext Markup Language* |
| IP | *Internet Protocol* |
| CPF | Cadastro de Pessoa Física |
| DNS | *Domain Name System* |
| HTTP | *Hypertext Transfer Protocol* |
| MIME Types | *Multipurpose Internet Mail Extentions Types* |
| JPEG | *Joint Photographic Experts Group* |
| XML | *Extensive Markup Language* |
| W3C | *World Wide Web Consortium* |
| ECMA | *European Computer Manufacturers Association* |
| PHP | *Hypertext Preprocessor* |
| Perl | *Practical Extraction and Report Language* |
| WHATWG | *What Working Group* |
| ARP | *Address Resolution Protocol* |
| MAC | *Media Access Control* |
| iOS | *iPhone Operational System* |

# INTRODUÇÃO

Em alguns anos, nossos arquivos e documentos estarão armazenados na nuvem: a tendência é que seja muito mais vantajoso não armazenar nada em nossos dispositivos pessoais. Enquanto este processo tem vários benefícios como a liberdade para acessar suas coisas em qualquer lugar, estes dados acessados pela internet nunca serão tão rápidos quanto acessá-los em seu computador. Não importa o quão rápido for a conexão, ela nunca será páreo para a leitura de dados em um disco rígido. Se por um lado o armazenamento está caminhando em direção a nuvem, com cada vez menos informações sendo armazenadas nos dispositivos dos usuários, em pequena escala (especialmente com os navegadores) diversas informações estão sendo armazenadas nos computadores clientes.

Este movimento vai contra a tendência do armazenamento na nuvem e ocorre por diversos motivos, como: menor tempo de resposta ao usuário, redução da carga de trabalho do servidor e maior acessibilidade.

Com a evolução da internet, as ferramentas de desenvolvimento utilizadas pelos programadores foram se tornando cada vez mais avançadas. Desde 2004 está em processo de desenvolvimento o HTML5 (Hypertext Markup Language 5), a quinta versão da principal linguagem de desenvolvimento para sites. Com ela, diversas funcionalidades que foram incluídas visando estas tendências, a nuvem e o armazenamento de informações nos computadores clientes. O LocalStorage (também conhecido como HTML5 Storage), que é a capacidade que a linguagem fornece para que os servidores armazenem dados nos navegadores e computadores clientes de forma antes nunca abordada, admite que aplicações sejam executadas de forma mais eficiente e permite aos usuários experiências novas. Permite, por exemplo, que não seja mais necessário estar conectado na internet para acessar o seu webmail.

Mas existem diversos fatores que devem ser analisados com o devido cuidado ao se desenvolver uma página com tais funcionalidades e, frequentemente, não são considerados com a devida importância.

Este trabalho tem por objetivo identificar os principais problemas de segurança associados ao LocalStorage, de forma a demonstrar como estes podem

ser explorados por eventuais atacantes e o quanto isso afetará na segurança da internet e na integridade dos usuários.

Inicialmente, no primeiro e segundo capítulo, serão abordados alguns conhecimentos importantes para que possamos entender o funcionamento da web, do HTML e da troca de informações entre clientes e fornecedores. Entenderemos também de onde surgiu o conceito de armazenamento no lado cliente da comunicação, compreendendo os cookies e o LocalStorage.

Em seguida, no terceiro capítulo, serão apresentadas as principais falhas e vulnerabilidades de segurança do HTML5 Storage, de forma a mostrar como um atacante poderia utilizá-las. No quarto capítulo, faremos uma análise de caso em um sistema que foi atacado, entendendo quais das falhas apresentadas foram exploradas.

No quinto capítulo será ilustrada a importância do cuidado com estas vulnerabilidades através de um exemplo real de uma aplicação. Por fim, abordaremos as principais medidas que devem ser utilizadas, visando proteger o sistema contra este tipo de falhas e, na conclusão, será apresentada uma breve análise sobre os impactos que estas vulnerabilidades apresentam, de forma a avaliar as vantagens e desvantagens de se utilizar o recurso e qual a melhor forma para protegê-lo, mitigando os riscos.

# 1 NOÇÕES SOBRE A INTERNET

Para desenvolver os conhecimentos e podermos nos aprofundar nos conceitos de armazenamento local de dados, é necessário conhecermos um pouco sobre o funcionamento da internet como um todo. Devemos conhecer os elos que compõem a internet e suas funções dentro dela. Vamos começar definindo a navegação na internet da forma que é utilizada hoje.

## 1.1 Sites, clientes, servidores e linguagens de programação

Um site é composto por vários arquivos que contém códigos. Estes códigos são escritos através de linguagens de programação e guardam informações que serão convertidas em conteúdo relevante para o usuário, quando interpretados por um *navegador*. Os arquivos ficam armazenados em um servidor na internet e são enviados para os “clientes” (computadores dos usuários) quando uma página é solicitada.

Para que sejam construídos estes arquivos, são utilizadas diversas linguagens de programação. Uma linguagem de programação é uma ferramenta utilizada para expressar instruções para um computador. São regras sintáticas e semânticas usadas para definir programas, permitindo especificar exatamente sobre quais dados o computador vai atuar, como serão armazenados e transmitidos sobre as mais diversas circunstâncias. Tradicionalmente, as instruções são escritas em arquivos-fonte que são “compilados”, ou seja, traduzidos em uma linguagem que permita ao computador entende-las. Mas, em se tratando da internet, temos o conceito de linguagens que são executadas no computador cliente, que atuam em conjunto com linguagens compiladas e executadas nos servidores.

As linguagens utilizadas para a escrita de sites são o HTML (Hypertext Markup Language, define a estrutura das páginas da web), o CSS (Cascading Style Sheets, utilizado para atribuir elementos e definições visuais aos elementos do HTML) e o JavaScript (utilizado para realizar validações, animações e diversas outras funções – inclusive o LocalStorage, em conjunto com o HTML5). Estas linguagens, diferente das compiladas, são escritas em arquivos e interpretadas pelos navegadores (como o Internet Explorer, Mozzila Firefox ou Google Chrome), transformando os códigos em conteúdo relevante para os usuários.

Como explicado acima, para que seja possível acessar os arquivos que contém estes códigos em qualquer hora e lugar no mundo, estes devem estar armazenados em um computador que, por sua vez, deverá ter um acesso à internet e ser capaz de identificar solicitações de outros computadores para entregar estes arquivos. Este computador é chamado de *servidor web* ou *servidor de hospedagem*.

## 1.2 A comunicação via IP

Mas como é que o computador do usuário sabe como encontrar o servidor web e quais arquivos solicitar? Todos os computadores conectados à internet possuem um endereço, que é conhecido *como endereço de IP (Internet Protocol)*. É uma sequencia de números (e caracteres hexadecimais, dependendo da versão do protocolo) que identifica de forma única um computador na internet. Fazendo uma analogia, o endereço de IP é como um número de CPF (Cadastro de Pessoa Física, o registro individual que cada brasileiro possui): assim como este primeiro associa um conjunto de informações a um indivíduo e permite que ele seja reconhecido unicamente no sistema do Brasil, o IP identifica um computador na internet e permite que outros o encontrem.

## 1.3 Domínios e DNS

Não é viável que o acesso aos sites disponíveis na internet seja realizado através de números cumpridos e difíceis de memorizar. Por este motivo, foi criado um sistema de nomenclaturas, o domínio. Conforme definição descrita pelo Registro.br, “domínio é um nome que serve para localizar e identificar conjuntos de computadores na Internet. O nome de domínio foi concebido com o objetivo de facilitar a memorização dos endereços de computadores na Internet.”. Como exemplo, podemos citar o acesso ao site do Google. O servidor do Google (ou pelo menos um dos muitos que eles dispõem) é identificado pelo endereço de IP 74.125.67.103. Contudo, ao digitar o endereço “www.google.com.br”, o nome é traduzido neste número, o qual é solicitado na web.

A resolução destes nomes de domínio, ou seja, a transformação de “www.google.com.br” para 74.125.67.103 (conforme nosso exemplo) é feita pelo DNS (Domain Name System), uma grande base de dados hierárquica distribuída

pela própria internet e que guarda informações para a tradução de nomes em endereços de IP e vice-versa.

## 1.4 Simplificando tudo

Podemos construir um panorama para exemplificar os conceitos apresentados até agora: O Sr. Fulano deseja acessar o site da FATEC. Ele abre o seu navegador e digita “www.fatec.br”. O seu computador então envia uma mensagem para um servidor de DNS, o qual irá resolver o nome de domínio solicitado em um endereço de IP. Em seguida, uma mensagem é enviada para este endereço, solicitando os arquivos do site da FATEC. O servidor da FATEC ao receber esta mensagem responde devolvendo os arquivos, os quais serão interpretados pelo navegador do Sr. Fulano e transformarão os códigos existentes nestes em conteúdo relevante.

## 1.5 Cookies

Agora que entendemos como funciona a comunicação na internet, digamos que o Senhor Fulano quer comprar um livro no site X. Ele entra no site, escolhe o título que deseja e faz a transação financeira. Após vários dias, o livro que ele comprou chega a sua residência. Então, no outro dia o Senhor Fulano deseja comprar um segundo livro. Ao entrar novamente no site X, ele é surpreendido com uma mensagem de boas vindas (do tipo “Olá, Senhor Fulano”) e uma lista de livros recomendados, do gênero que o Fulano gosta de ler. Como isso é possível, se o Senhor Fulano não informou nada ao site quando acessou pela segunda vez?

Esse é um dos exemplos mais comuns de utilização dos *cookies*. Conforme definição dada pela Mozzila Foundation, “um cookie é um pequeno texto que os sites podem enviar aos navegadores, anexado a qualquer conexão. Nas visitas posteriores o navegador reenvia os dados para o servidor dono do cookie” [3]. Desta forma, quando o Senhor Fulano comprou o seu livro, o servidor enviou para o navegador dele um pequeno arquivo com informações que identificam o navegador do Senhor Fulano de forma única. Então, no segundo acesso ao site, o navegador enviou este cookie juntamente à solicitação das páginas desejadas, identificando que o solicitante era o Fulano. Ao receber estas informações, o servidor utilizou-as para “lembrar” das preferências do usuário e tornar a navegação personalizada.

Contudo, os cookies possuem algumas características que os tornam indesejáveis, em alguns casos. Incondicionalmente, eles são inclusos em todas as requisições HTTP (Hypertext Transfer Protocol, todas as requisições de páginas aos servidores) e, portanto, deixam as aplicações na web mais lentas. Além disso, a não ser que a aplicação web seja completamente protegida por criptografia, os cookies são enviados sem que haja nenhum tipo de proteção em seus dados, expondo o seu conteúdo. Não se pode esquecer também que, como se tratam apenas de “ponteiros” para informar aos servidores quem é o usuário que está solicitando os arquivos, os cookies são limitados a 4KB de informações – suficiente para deixar as aplicações mais lentas e, contudo, não é uma quantidade realmente útil de informações.

## 1.6 O Local Storage

Buscando soluções para estes problemas e novas possibilidades de aplicações para a internet, surge com o HTML5 o Local Storage.

Resumidamente, o Local Storage é uma forma para as páginas armazenarem pares de chaves / valores localmente, isto é, na máquina cliente, dentro dos navegadores. Assim como os cookies, os dados persistem mesmo após o usuário sair do site ou até mesmo do navegador [7]. Porém, ele se diferencia de forma que os dados armazenados não são transmitidos para os servidores remotos (a não ser que sejam enviados manualmente). O acesso aos dados é feito via JavaScript, caso o navegador utilizado suporte este recurso.

# 2 AS LINGUAGENS DA INTERNET

## 2.1 Hypertext Markup Language

O HTML surgiu em meados de 1990. De lá, passou por inúmeras mudanças, que transformaram a experiência do usuário e as aplicações desenvolvidas para a web. Basicamente, é uma linguagem formada por marcações (também chamadas de tags). Para cada elemento que se deseja exibir, é necessária a escrita de uma marcação específica de código, e cada uma destas é renderizada de maneira única. Estas marcações se organizam de forma hierárquica e podem possuir diversos atributos que as caracterizam de forma ainda mais específica.

Uma **marcação** é caracterizada por uma palavra entre os sinais “<” e “>”. Alguns exemplos de marcações:

<head> e </head> (marcação para determinar conteúdo do cabeçalho do site)
<body> e </body> (marcação do conteúdo do “corpo” da página)
 e  (marcação para abrir um bloco de conteúdo)

Como se trata de uma linguagem com propriedades hierárquicas, deve haver códigos que sinalizam o fim da marcação, associando todo o conteúdo dentro dela como marcações-filhas. Conforme o exemplo acima, para todo “ ” aberto, deverá ter um “ ” fechando. Estas marcações podem ter atributos, os quais os tornam acessíveis através de outras linguagens ou propriedades dos navegadores. Por exemplo:

```
<div id=”exemplo” class=”bloco”>
        Exemplo de div
</div>
```

Neste exemplo, a marcação  é identificada unicamente por “exemplo” e é da classe “bloco”. Estas propriedades específicas tornam possível atribuição de características visuais específicas utilizando o CSS (Cascading Style Sheets, ou Folhas de Estilo em Cascata, linguagem criada para a edição e atribuição de elementos visuais às marcações do HTML) ou ainda ativar uma função específica em Javascript (iremos abordá-lo em breve).

Em todas as linguagens de programação é possível fazer comentários no código, instruções que não serão lidas pelo interpretador (seja um compilador ou um navegador), introduzidas de forma facilitar o entendimento do código ao ser lido. Cada linguagem utiliza do seu próprio sistema de comentários. Para o HTML, é utilizado as marcações “<!--“ para abrir um comentário e “-->” para fechá-lo. No Javascript, a marcação é “/*” para abrir e “*/” para fechá-lo ou através de uma única marcação “//”.

Para ilustrar melhor o sistema de marcações do HTML, vamos examinar este exemplo de página simples, utilizando de comentários:

```
<html>
<!-- toda página de html deve iniciar com esta tag -->
    <head>
    <!-- Abre o cabeçalho da página, área com informações (em sua maior parte) invisíveis ao usuário -->
        <title>Exemplo de Código</title>
        <!-- titulo da página -->
    </head><!-- fecha o cabeçalho da página -->
    <body><!-- Abre o corpo da página, área na qual o conteúdo visível ao usuário será inserido -->
    <h1>Exemplo de página</h1>
    <!-- As marcações <h1>, <h2> e assim por diante marcam “títulos” na página – O conteúdo destas tags é renderizado com maior destaque -->
        <div id=”bloco”>Bloco de conteúdo</div>
        <!-- A marcação <div> abre um bloco de conteúdo. Ao colocar o atributo id=”bloco”, indentificamos esta <div> de forma única, permitindo associar um estilo CSS a ela ou posteriormente utilizar ações em Javascript -->
    </body> <!-- fecha a marcação do corpo da página -->
</html> <!-- fecha a página HTML -->
```

Poderíamos inserir uma imagem neste código através de sua marcação específica: a tag <img>. Mas antes de examinar este elemento do HTML, temos que entender os MIME Types (Multipurpose Internet Mail Extentions Types).

## 2.2 MIME Types e o Surgimento do HTML5

Toda vez que o seu navegador faz uma requisição de página a um servidor (com suas respectivas marcações, como mencionado acima), antes de enviar os códigos contidos nos arquivos este segundo envia determinados “cabeçalhos”, invisíveis aos usuários (exceto por meio de ferramentas de desenvolvimento). A importância destes cabeçalhos é que eles irão dizer ao navegador como interpretar os códigos que serão enviados em seguida pelo servidor. E o mais importante desses cabeçalhos é chamado de *“Content-Type”*, e se parece com isso:

Content-Type: text/html

Esta marcação “*Content-Type*” (ou MIME type) é quem determina o que um elemento ou recurso realmente é e, consequentemente, como deve ser interpretado e/ou renderizado. Todos os elementos possuem um tipo MIME, sejam imagens (image/jpeg ou image/png, etc), javascript, css. Pode-se dizer até que a web é baseada nos MIME types. [4]

Continuando a entender as marcações do HTML, se utilizamos a marcação <img> para referenciar uma imagem teremos um código parecido com este:

```
<img src=”pasta/imagem.jpg”  />
```

Ao enviar este código navegador envia simultaneamente o seu MIME Type que, no caso, é “image/jpeg” (imagem, com codificação JPEG – Joint Photographic Experts Group).

A importância de evidenciar estas marcações e os MIME Types é entender como funciona o *gerenciamento de erros draconiano* (ou Draconian Error Handling).

Desde o surgimento do HTML, os navegadores sempre foram tolerantes com os erros nas marcações. Por exemplo, se você criar uma página, mas esquecer de utilizar a marcação </head> para fechar o cabeçalho desta mesma, os navegadores irão renderizar a página do mesmo jeito. De alguma forma, os navegadores

conseguem lidar com este tipo de erro e continuar seu trabalho sem que sejam exibidos erros ao usuário que as visualizam. O fato de páginas serem criadas utilizando de marcações “defeituosas” ou “quebradas” faz com que páginas inteiras sejam criadas de forma errada, defeituosa e “quebrada”. Existem estimativas de que 99% das páginas existentes hoje na internet possuam ao menos uma marcação mal-escrita.

Para corrigir este problema, uma das versões do HTML (publicada em 1997) foi elaborada com o intuito de que as marcações fossem baseadas em XML (Extensive Markup Language, linguagem de marcações assim como o HTML, mas criada para descrever diversos tipos de dados) e, assim, não fossem tolerantes a estes tipos de erros mencionados no paragrafo acima. Páginas defeituosas simplesmente geravam erros “fatais” e não eram renderizadas. Este conceito de falhar totalmente após um simples erro (como por exemplo fechar a tag  com uma tag </dvi> ao invés de ) ficou conhecido como Gerenciamento de Erros Draconiano (o termo veio do líder Grego Draco, que instituiu a pena de morte para quaisquer infrações em suas leis, por menor que fossem).

Assim, quando o HTML foi reformulado como um vocabulário do XML e seu MIME Type passou a ser application/xhtml+xml, qualquer forma de erro nas marcações, mesmo que o esquecimento de uma entre as dezenas de marcações na página, resultaria em uma falha e consequente finalização da renderização com uma mensagem de erro aos usuários.

Apesar das inúmeras evoluções que esta atualização da linguagem trouxe, a ideia se tornou extremamente impopular, já que a grande maioria da web era construída sobre páginas defeituosas. Dessa forma, os autores e desenvolvedores ignoraram esta nova regra, mas não ignoraram a nova sintaxe. As páginas eram escritas utilizando de novas marcações e funcionalidades, mas o MIME Type text/html foi mantido. E ainda hoje, milhões de páginas são escritas desta maneira.

Em junho de 2004, diversos workshops e convenções realizadas entre os maiores fabricantes de navegadores e a W3C (World Wide Web Consortium, consórcio responsável pela regulamentação da web) ocorreram com o intuito de desenvolver uma evolução do HTML (que estava em sua quarta versão), para

resolver diversos destes problemas e agregar novas funcionalidades na linguagem. Os princípios que foram relevantes para a elaboração da nova versão abordavam:

- Compatibilidade com as versões anteriores
- Gerenciamento de erros bem-definidos
- Os usuários não deveriam ser expostos aos erros dos autores
- Uso prático
- Integração definitiva com scripts de outras linguagens
- Padronização da linguagem em diversas plataformas
- Visualização publica

O processo não foi tão rápido e nem simples. Mas a partir disso, surge o HTML5.

O HTML (mesmo em suas versões anteriores, contudo acentuadamente na sua última versão) tem alta compatibilidade para trabalhar com outras linguagens. Esta capacidade fez e faz com que a linguagem seja cada vez mais poderosa, permitindo que formas de aplicações antes impensáveis sejam construídas. Como foi explicado acima, chegou a hora de entender um pouco esta compatibilidade entre o HTML e a principal linguagem que vem permitindo as maiores revoluções proporcionada pelo HTML5: o Javascript.

## 2.3 O JavaScript

O JavaScript foi criado em 1995 pela Netscape, com o objetivo de permitir a manipulação dinâmica dos elementos de uma página HTML apresentados nos navegadores. Por conta do grande sucesso, logo em 1996 já havia uma versão do JavaScript para o Internet Explorer. Com o objetivo de padronizar a linguagem, seus direitos autorais foram entregues em 1997 para a ECMA (European Computer Manufacturers Association) e a partir daí as empresas adequaram suas linguagens ao padrão estabelecido. Por esta razão, o Javascript também é conhecido como ECMAScript.

Os scripts dessa linguagem são incluídos em páginas HTML através de três formas, conforme os exemplos abaixo:

1. Colocando as instruções entre as tags <script> e </script>

```
<html><head></head>
    <body>
        <script language=”Javascript 1.5”>
        <! –-
        //O número da versão da linguagem é opcional
        var str = “Olá, Fulano!”;
        document.write(str);
        -->
        </script>
    </body>
</html>
```

Neste exemplo, foi criada uma variável str e atribuído um valor a ela. Em seguida, o conteúdo da variável é exibido na página através de uma função (conjunto de instruções da linguagem que existem para referenciar certos procedimentos pré definidos. No caso, a função utilizada foi a *document.write())*.

2. Inserindo manipuladores de eventos dentro de tags HTML

```
<body onLoad=”umaFuncao();” >
```

A propriedade “*onLoad*” inserida na marcação body é uma manipuladora de eventos, que será ativada no momento em que a marcação for carregada. Ao ativar este manipulador, o mesmo irá executar “umaFuncao()”. Este mesmo exemplo foi utilizado acima neste capítulo.

3. Inserindo o código dentro de um arquivo e referenciando-o em uma tag “<script>”

```
<script src=”scriptExterno.js”></script>
```

O atributo “src” da marcação <script> indica a localização do arquivo que contém os códigos, os quais são “importados” para dentro da página.

## 2.4 Cookies

Cookies são pequenos pedaços de informação gerada por servidores web e armazenados nos computadores dos usuários, para posteriores acessos. Eles são

embutidos nos códigos HTML e fluem entre os clientes e servidores. Os cookies são implementados para permitir a customização das informações. Por exemplo, podem ser utilizados para personalizar um motor de buscas na web, para permitir que os usuários participem de um concurso, limitando para apenas uma vez ou ainda para armazenar lista de compras feitas pelos usuários durante uma visita a uma loja virtual. Os servidores web automaticamente ganham acesso aos cookies quando os computadores clientes conectam a eles.

Em 1994, programadores da Netscape (uma das primeiras fabricantes de navegadores) enfrentavam um problema. Naquele momento da web, qualquer visita a um site seria como a primeira, com nenhuma forma automática para registrar que um visitante havia passado lá antes. Qualquer transação comercial deveria ser manejada do começo ao fim em uma única visita, e os usuários deveriam passar por todas as fases do processo repetitivamente. Era como visitar uma loja onde o atendente tinha problemas com sua memória.

A solução encontrada foi a de que os sites criavam e armazenavam um pequeno arquivo nos computadores dos usuários, mantendo registros do que foi realizado durante a visita. Os desenvolvedores da tecnologia chamaram esta funcionalidade de "*persistente client state object*", ou objeto persistente de estado do cliente. Quando as máquinas passaram pequenos bits de código para propósitos de identificação, os programadores chamaram os dados trocados de "*magic cookies*", ou Cookies Mágicos.

Foi um ponto chave na história da internet. Os cookies transformaram a web de um lugar para visitas aleatórias em um ambiente rico e cheio de interação. Dessa forma, os cookies alteraram a forma como navegamos na internet.

Os cookies são estruturas de texto simples. São normalmente encontrados nas áreas de cache do navegador ou em arquivos temporários do sistema operacional. Cada registro de cookie é composto de um par de nome-valor contendo as informações desejadas, uma data de expiração na qual o cookie deverá ser removido e o domínio do qual ele está associado. Para ler ou escrever entradas em um arquivo de cookie, é utilizado o JavaScript. Cada nome de domínio na web pode possuir um arquivo de cookie associado e cada cookie pode ter diversas entradas de dados. No servidor, a leitura e escrita dos cookies recebidos e enviados são feitas

através de uma linguagem específica, como PHP (Hypertext Processor) ou Perl (Practical Extraction and Report Language).

Todas as páginas HTML possuem uma propriedade que é acessada via JavaScript, chamada documento.cookie. Esta propriedade é utilizada para ler e obter as informações do mesmo. Para criar um cookie para um site, utiliza-se o seguinte script:

```
<script type=”text/javascript”>
        document.cookie = “valorteste1=Yes; expires=Fri, 13 Jul 2004
05:28:21 UTC; path=/”;
</script>
```

Todo o valor registrado é fornecido como um texto entre aspas, com os valores separados por ponto e vírgula – primeiro o nome-valor, em seguida a data de expiração no formato correto e por último o caminho onde o cookie está armazenado.

## 2.5 LocalStorage

As primeiras versões do HTML5 eram dotadas de um mecanismo de armazenamento chamado *Global Storage*. Em resumo, é um espaço de memória concedido pelo navegador no qual os sites podem armazenar dados persistentes que não necessitam ser enviados para o servidor. Os dados podem ser acessados por Javascript (ou ainda pelo Flash, tornando uma ótima ferramenta para o desenvolvimento de jogos). Para escrever ou alterar os dados, contudo, devem ser definidos os domínios com permissão para tais eventos.

As especificações determinavam que todos os domínios de nível igual ou menor (todas as subpastas do domínio) ao definido para o Global Storage como domínio principal poderiam acessar os dados armazenados. Mas os desenvolvedores não implementaram estes recursos e nunca suportaram o armazenamento de informações publicas que que não pertencessem ao domínio principal. De fato, o navegador da Mozzila Foundation, o Firefox, bloqueou o acesso a qualquer outro domínio que não o especificado como principal.

Após os navegadores implementarem as suas versões do Global Storage e esta passar a ser utilizada em grande escala, o WHATWG (What Working Group) modificaram as especificações e substituíram-no pelo LocalStorage, removendo a capacidade de especificar diferentes domínios. Todos os dados armazenados são automaticamente associados ao domínio no qual os scripts estão rodando.

Os scripts utilizados para gerenciar os objetos são muito parecidos com os utilizados para os Cookies, diferenciando apenas na persistência e no escopo. O LocalStorage é utilizado para armazenamento por longos períodos e seus dados permanecem gravados mesmo se o navegador for fechado. Os dados armazenados pelos cookies, em contraponto, expiram ao encerrar a sessão.

Outra vantagem sobre os cookies é que o LocalStorage é acessível em todos os navegadores, enquanto os primeiros estão confinados nos navegadores os quais foram criados.

Por se tratar de uma funcionalidade muito recente, existem diversos navegadores que não suportam o LocalStorage. Dessa forma, antes de utilizar o recurso, é necessário checar se há o suporte. Este procedimento pode ser realizado através de JavaScript:

```
<script type=”text/javascript”>
if (typeof localStorage () == 'undefined') {
        alert (‘Sem suporte a LocalStorage');
}
</script>
```

Se o navegador suportar, basta utilizar os scripts para manipular os dados:

```
localStorage.setItem ("nome", "João da Silva");
// insere um par de registros no LocalStorage de chave (“nome”) e valor (“João da Silva”)

var nome = localStorage.getItem ("chave1");
  //atribui o valor do registro “chave1” a uma variável chamada nome

localStorage.removeItem ("nome");
```

/ / exclui o item correspondente do banco de dados

Estes dados ficam armazenados em pequenos bancos, como segue:

| Chave | Valor |
|---|---|
| **Nome** | João da Silva |
| **Idade** | 18 |
| **Chave1** | Valor1 |
| **Chave “n”** | Valor “n” |

Figura 1 – Exemplo de Chaves / Valores do LocalStorage

Portanto, ao ler o comando localStorage.getItem(“nome”), por exemplo, o navegador iria até a base do LocalStorage e iria ler o valor correspondente à esta chave que, no caso, é “João da Silva”. Este valor pode ser atribuído a uma variável ou utilizado para qualquer outro propósito que o desenvolvedor deseje.

## 2.6 As Ferramentas Utilizadas

Além do JavaScript, existem navegadores que permitem que os dados sejam visualizados diretamente, através de consoles e ferramentas de desenvolvimento.

Os navegadores mais recentes possuem a capacidade de serem expandidos através de plug-ins e extensões. Estes pequenos programas que são “anexados” ao navegador ao serem instalados pelos usuários são capazes de realizar diversas funções, como melhorar os sistemas de busca em páginas, rodar jogos localmente, alterar temas e cores e, o principal para este trabalho, oferecer alternativas para a visualização do código fonte e dos elementos JavaScript de uma página.

Neste projeto, foi utilizado o navegador Google Chrome e sua ferramenta nativa para desenvolvimento, que permite visualizar os elementos de HTML, CSS e Javascript das páginas de qualquer site.


Figura 2 - Ferramentas de programação do Google Chrome

A imagem acima é a ferramenta de desenvolvimento do Google Chrome. Este painel pode ser acessado através do botão “Personalizar e controlar o Google Chrome”, localizado ao lado da barra de endereços (barra onde se digita os sites a serem acessados). Neste botão, basta seguir o menu “Ferramentas” -> “Ferramentas do Desenvolvedor”. O console se abre na parte inferior da tela e seu tamanho é personalizável.

A barra de ferramentas no topo lista os instrumentos disponíveis para o desenvolvedor. Para nosso trabalho, iremos utilizar a aba Elements (Elementos), Resources (Recursos) e Console.

Na primeira aba Elementos (ilustrada na imagem acima), o painel do lado esquerdo mostra o código-fonte da página em tempo real, com todos os links e classes, organizados hierarquicamente conforme os códigos HTML. O painel do lado direito é a ferramenta para controle do CSS e os atributos visuais dos elementos, que são selecionados no painel anterior.

A segunda aba, Recursos, possui também dois painéis:

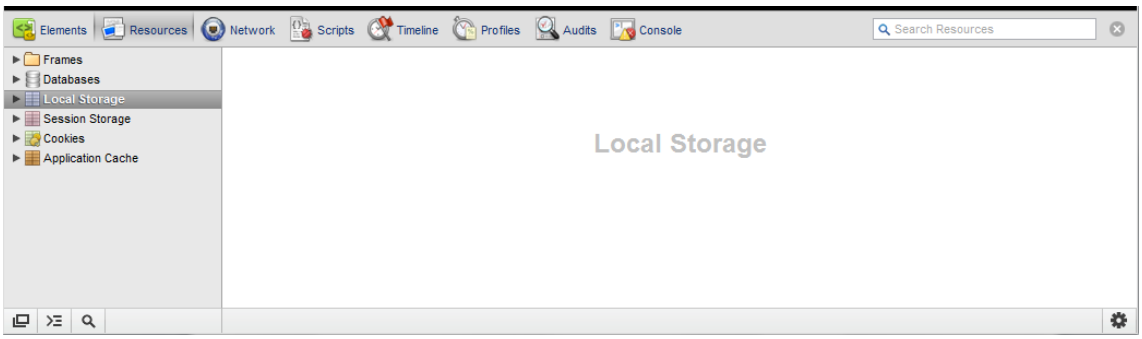



Figura 3 – aba “Recursos” das Ferramentas do Desenvolvedor

No lado esquerdo, há um menu no qual podemos selecionar as informações que queremos visualizar, as quais serão exibidas no painel do lado direito.

A última aba importante para nós, o Console, é um terminal no qual podemos escrever comandos e scripts em JavaScript e visualizar as atualizações em tempo real na página.

## 2.7 Como Visualizar o LocalStorage

Para visualizar os dados armazenados em um computador cliente através das ferramentas de desenvolvedor do Chrome, basta abrir o painel através do caminho mencionado acima, na aba recursos selecionar a opção Local Storage.

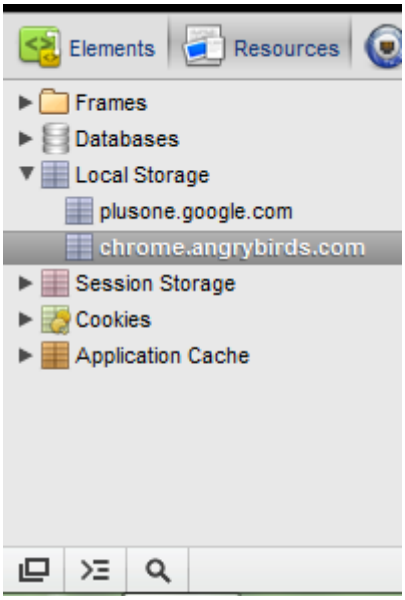



Figura 4 - Página de recursos das Ferramentas de Desenvolvedor

Ao selecionar esta opção um submenu irá se abrir exibindo as bases existentes associadas a página em exibida no navegador. No exemplo acima, existem duas bases listadas, referentes às duas páginas abertas no momento: plusone.google.com (referente a página plusone.google.com.br) e chrome.angrybirds.com (referente página chrome.angrybirds.com). Ao selecionar a base chrome.angrybirds.com, no painel ao lado direito são exibidas as seguintes informações:

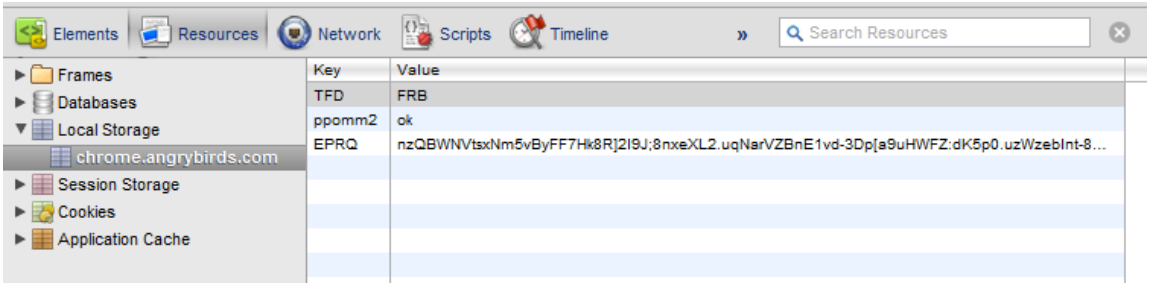



Figura 5: Exibição do LocalStorage na página de Recursos

Conforme ilustrado anteriormente, o painel mostra a relação de chaves/valores existentes na base. Por exemplo, se acessarmos a chave “TFD “através do comando localStorage.getItem(“TFD”), teríamos como retorno o valor “FRB”.

# 3 AS VULNERABILIDADES DO LOCALSTORAGE

Uma vez que conhecemos o funcionamento da web e do LocalStorage, podemos nos aprofundar no objetivo deste trabalho. Neste capítulo, serão listadas as principais vulnerabilidades do LocalStorage, evidenciando o seu funcionamento e mostrando exemplos de ataques utilizados para obter informações dos usuários.

## 3.1 Fraudes de DNS

Como já vimos anteriormente, os dados armazenados no LocalStorage só poderão ser acessados e controlados localmente, ou seja, diretamente no computador cliente (através de uma ferramenta de desenvolvimento como a do Google Chrome) ou pelo servidor proprietário da página. Por exemplo, se o Gmail criar um LocalStorage com informações quaisquer, estes dados só poderão ser acessados localmente no computador que estiver acessando o site ou pelo servidor do Gmail, identificado pelo domínio mail.google.com.

Contudo, por causa do potencial do DNS para ataques de fraude, não é possível garantir que um servidor que alega pertencer a um domínio é realmente daquele domínio [14].

Quando solicitamos uma página ou qualquer outra informação através de um navegador, uma mensagem é enviada a um servidor DNS que irá responder com o endereço IP correspondente e desejado. Entretanto, se este servidor DNS estiver corrompido ou possuir programas mal intencionados, podemos receber um endereço diferente do correto, no qual um computador utilizado por um atacante estará se passando pelo servidor que inicialmente desejamos acessar. Dessa forma, o computador atacante terá as ferramentas e permissões necessárias para acessar o LocalStorage, manipulando e obtendo as informações como bem entender.

Para resolver este problema, é necessário fazer com que as informações trocadas entre o cliente e o servidor original sejam criptografadas, ou seja, fiquem codificadas de forma que se alguém que não pertença a conversa (como, por exemplo, um computador malicioso informado por um servidor DNS) receber os dados, não vai conseguir tirar nenhum proveito.

Ao acessar um site através do navegador, o computador cliente “pergunta” ao servidor DNS o número de IP correspondente ao endereço do site informado, para que o navegador possa enviar uma nova solicitação ao servidor desejado.

Na fraude de servidor DNS, quando a requisição inicial é enviada o servidor malicioso envia um endereço de IP diferente do real, levando o computador cliente a solicitar o site a um computador diferente do que deveria ser, no qual frequentemente é encontrado uma aplicação para roubar os dados do usuário ou instalar programas maliciosos.

Como consequência, o usuário muitas vezes acredita que realmente está no site desejado, quando na verdade está em um site “pirata”. Este tipo ataque pode ser muito perigoso quando são falsificados sites importantes como bancos e instituições financeiras. Frequentemente, o site falso terá a mesma interface do site real, aumentando a sensação do usuário de que não há nada de errado.

Normalmente, estes sites são criados para que os próprios usuários digitem seus dados pessoais, fazendo posterior uso das informações. Este tipo de fraude ainda pode ser utilizado para que o site acessado tenha acesso ilimitado aos dados do LocalStorage, o que pode ser ainda mais perigoso, pois mesmo que o usuário seja atento e não digite nenhuma informação relevante, a aplicação no servidor terá acesso a estes dados. Uma das formas mais comuns de fraudar um servidor DNS é ataque chamado “*Man In The Middle*”, que consiste na interceptação de troca de pacotes através de computador malicioso posicionado entre outras duas máquinas e manipulando o conteúdo recebido. Este método é baseado na técnica de ARP-Poisoning (envenenamento de ARP - *Address Resolution Protocol*).

Para se conectar em uma rede, além do endereço de IP que irá identifica-lo, o computador deverá ter também um endereço de MAC (*Media Access Control*), o qual é (supostamente) unicamente atribuído a cada interface de comunicação do computador. O MAC Address é utilizado na camada de enlace, ou seja, atua diretamente no endereçamento físico das placas de rede. Quando um pacote é enviado entre um computador e outro, a aplicação que envia tem apenas que saber o endereço de IP do destinatário, deixando o endereçamento de MAC para a placa de rede.

Para realizar o endereçamento de MAC, é utilizado o protocolo ARP. Basicamente, o protocolo cria tabelas com endereços de MAC e seu IP correspondente e armazena nos computadores, *switches* e roteadores. Quando uma máquina não possui o MAC address desejado na rede, ele envia uma mensagem solicitando que todos os membros da rede se identifiquem, para que se o computador que possui aquele MAC existir, ele seja localizado.

Contudo, por se tratar de um protocolo que atua em uma camada física, ele não é orientado a conexão. Isso significa que um computador não consegue “se lembrar” se enviou uma solicitação de identificação ou não. Dessa forma, para enganar a tabela de um computador A, tudo que o computador B tem que fazer é enviar uma mensagem no protocolo ARP dizendo “eu tenho o MAC Address C” e o computador A acatar a essa informação.

Portanto, o envenenamento de ARP é um tipo de ataque no qual uma falsa resposta é enviada a um computador 1 por um computador 2, convencendo-o seu falso endereço. O computador 3, dono do endereço falsificado por 2, não tem como saber que o redirecionamento ocorreu. [9]

A partir desta premissa, um computador que alega ser um servidor DNS na rede pode receber as requisições de páginas e encaminhá-las ao servidor DNS falsificado. Dessa forma, o Envenenamento de ARP é utilizado para realizar a falsificação de DNS através da técnica *Men-in-the-Middle*, encaminhando o computador solicitante a um servidor falso que poderá fazer uso dos dados do LocalStorage.

## 3.2 Ataques Entre Diretórios

Existem casos em que são instalados vários sites em um único servidor. Na verdade, isso ocorre com a maioria dos sites que existem na internet. Empresas de hospedagem geralmente colocam dezenas (e até centenas) de sites hospedados em um único servidor, compartilhando hardware, software, nomes e identificações. Isso faz com que diversos sites sejam acessados a partir do mesmo IP e, muitas vezes, mesmo nome DNS.

Como o LocalStorage valida o acesso aos dados pelo domínio, os outros sites hospedados neste servidor irão acessar estas informações sem nenhum problema. Esta é uma das mais graves falhas de segurança deste recurso e o uso do mesmo neste tipo de ambiente não é recomendado.

Digamos que no servidor A estão hospedados os sites B e C. Através de requisições JavaScript executadas pelo site C, é possível acessar arquivos e enviar comandos ao site B.

## 3.3 Acesso Não Autorizado a Dados

Um atacante com acesso físico (ou remoto) a uma máquina pode, através do próprio navegador, visualizar, acessar e alterar as informações presentes no recurso LocalStorage.

Por exemplo, um programa malicioso instalado no computador do usuário que tenha acesso aos dados no navegador poderia coletar os mesmos e enviar a um destinatário remoto. Neste caso, se um site armazenar informações relativas à conta de um usuário em um determinado serviço através do LocalStorage e estas forem lidas por um programa malicioso, a conta deste estaria seriamente exposta a invasões.

Há também a possibilidade de um atacante malicioso, ao obter acesso a um computador, copiar as informações do LocalStorage no navegador e inseri-las manualmente em um segundo computador. Se estas informações forem relevantes e validarem a conta deste usuário em um sistema, isto poderia agregar diversos riscos.

Para utilizar-se dos dados armazenados no LocalStorage, tudo que o atacante tem que fazer é abrir um navegador que tenha suporte a ferramentas de desenvolvimento (o Google Chrome, por exemplo) e visualizar os dados, conforme explicado no capítulo anterior.

Ele pode armazenar estes dados e posteriormente inseri-los em um navegador de outro computador, fazendo com que o site que armazenou os dados não perceba que quem está usando é um atacante. Dessa forma, mesmo que as informações armazenadas no localstorage não sejam relevantes, o agressor pode

usá-las para descobrir outras informações que comprometam a segurança do usuário.

# 4 ANÁLISE DE CASO

Neste capítulo, será analisado um caso conhecido de exploração de uma vulnerabilidade do LocalStorage. A análise ocorrerá sobre um aplicativo desenvolvido para diversas plataformas e que possui uma versão para browser, o *Angry Birds*. É um jogo bastante popular, com recursos especiais quando executado através do Google Chrome.

## 4.1 Cenário da análise: O jogo Angry Birds

O Angry Birds é um jogo de estratégia desenvolvido pela Rovio mobile, lançado para o iOS (iPhone Operational System) da Apple em 2009. Após a seu lançamento, a Rovio Mobile portou o jogo para outras plataformas e desenvolveram uma versão para o Google Chrome que pode ser baixada através da Chrome Web Store (https://chrome.google.com/webstore/detail/aknpkdffaafgjchaibgeefbgmgeghloj?hl=pt-br&hl=pt-BR&brand=CHLG).


Figura 6 - Tela Inicial do Jogo Angry Birds

O jogo se passa em um mundo virtual, no qual os “porcos” roubam ovos dos “pássaros” para comer. Os pássaros ficam muito irritados e se juntam com diversas raças para se vingar dos porcos e recuperar os ovos.

Existem duas áreas importantes no jogo: a “base”, do lado esquerdo do cenário no jogo, no qual ficam o “estilingue” e os pássaros, e a base inimiga, do lado direito do cenário, no qual ficam os “porcos”; Geralmente estes estão protegidos por madeiras, ferros e outros materiais.

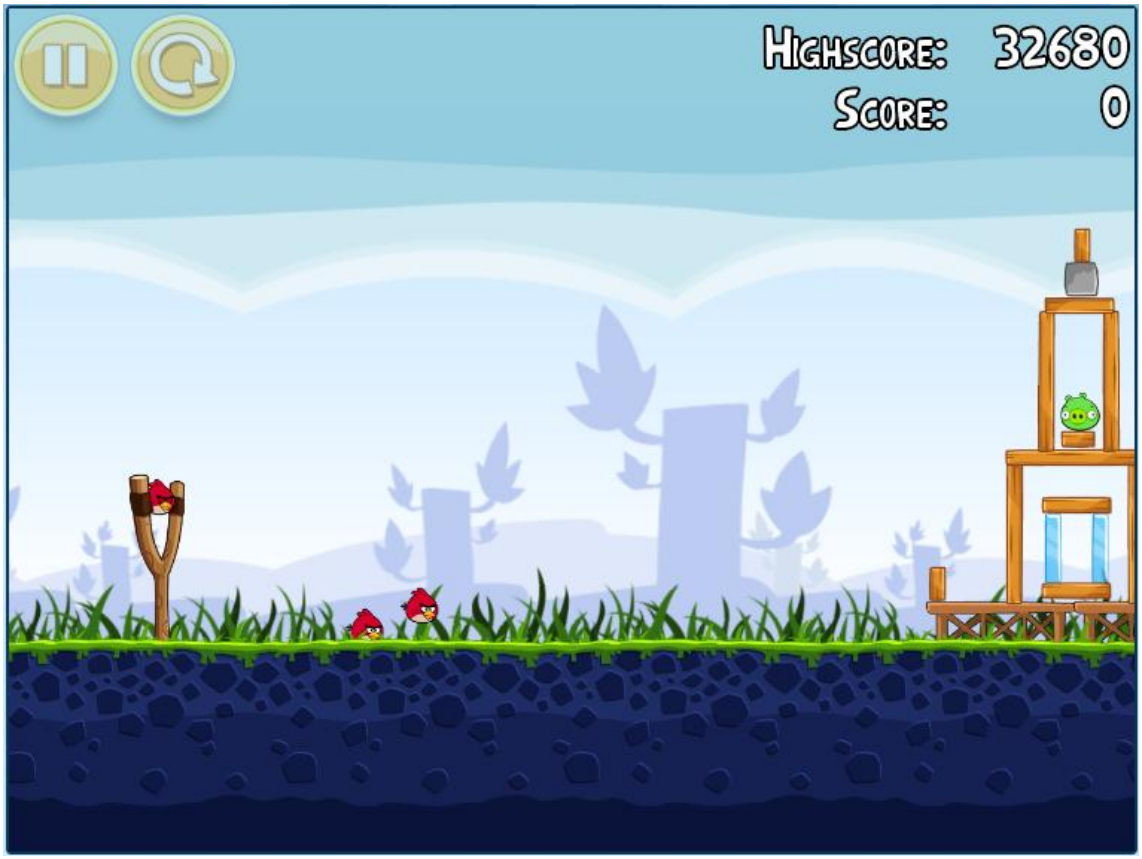



Figura 7 - Tela do Jogo Angry Birds

O objetivo do jogo é arremessar “pássaros” contra os inimigos através do estilingue, de forma a acertá-los e destruí-los. Quanto menos pássaros forem utilizados para se destruir todos os porcos, maior é a pontuação.

O jogo possui ao todo 70 níveis. Ao completar um nível, o jogador recebe uma pontuação que varia entre uma, duas ou três estrelas. É necessário pelo menos receber uma estrela de pontuação para liberar o próximo nível, e assim por diante até que todos os níveis estejam completos.

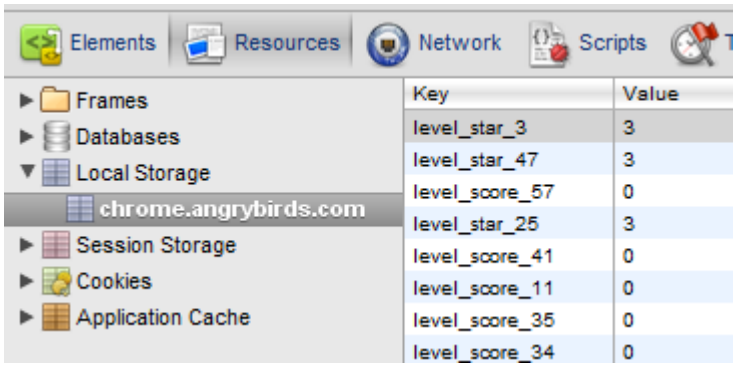
## 4.2 A Vulnerabilidade

Após vários dias do lançamento do Angry Birds para o Google Chrome, o jogo foi *hackeado* por Wes Bos. De acordo com um artigo no site thenextweb.com, o ataque de Bos foi desenvolvido graças a informações providas pela Rovio de que as informações do jogo são armazenadas utilizando o LocalStorage. [13]

A pontuação fica armazenada no LocalStorage através de um conjunto de chaves e valores:

| Key | Value |
|---|---|
| Star_level_1 | 3 |
| Star_level_2 | 3 |

O star_level_x representa o nível, e o valor correspondente a esta chave significa a pontuação recebida (1 para uma estrela, 2 para duas estrelas e 3 para três estrelas).




Armazenamento das informações de pontuação no localStorage

Baseado nesse cenário, um usuário que utilize o console para visualizar o LocalStorage pode alterar os valores manualmente, liberando todos os níveis com pontuação máxima.

## 4.3 Explorando a brecha

Todos os navegadores atuais permitem que sejam executados códigos JavaScript inseridos na barra de endereços. Dessa forma, além de alterar os valores

manualmente no console, o atacante pode também executar um comando através deste recurso e inserir os dados desejados no LocalStorage.

Neste caso, o código abaixo pode ser usado perfeitamente [6]:

```
javascript:
 var i = 0;
while (i<=69) {
    localStorage.setItem('level_star_'+i,'3');
  i++;
 }
window.location.reload();
```

Na primeira linha, o comando “javascript:” avisa o navegador de que estamos executando um comando e não solicitando um endereço. Na linha de baixo, é instanciada uma variável chamada i e atribuído o valor 0 a ela. Esta variável irá servir como um contador para a nossa função.

Em seguida, é executada a função while:

```
while (i<=69) {
```

Com este código, instruímos o navegador de que enquanto a variável “i” for menor ou igual a 69, ele deve continuar executando os códigos que seguem. Como atribuímos o valor 0 a esta variável, o código irá correr normalmente.

```
 localStorage.setItem('level_star_'+i,'3');
i++;
```

Na primeira linha, inserimos um valor no localStorage através da função anteriormente vista, a “localStorage.setItem()”. A chave atribuída é a ‘level_star_’+i. Durante a iteração, o navegador irá substituir a variável “i” pelo seu valor corrente. Isso significa que na primeira iteração da função while, esta chave terá o valor ‘level_star_0’. O segundo parâmetro da função determina que o valor correspondente a esta chave é ‘3’.

A segunda linha, ‘i++;’, faz um incremento em 1 no valor atual da variável “i”. Se a variável valia ‘0’ na primeira iteração, ao iniciar a próxima ela irá valer 1 e,

consequentemente, a chave que será adicionada ao LocalStorage nesta segunda iteração será ‘level_star_1’. Este ciclo se repete até que a variável “i” atinja um valor igual a 69.

A próxima e última linha:

```
window.location.reload();
```

Este comando recarrega a página corrente, fazendo com que o LocalStorage seja recarregado e, consequentemente, todas as chaves adicionadas na função anterior sejam ativadas, liberando todos os níveis do jogo para os usuário.

# 5 DEMONSTRAÇÃO PRÁTICA

Para melhor ilustrar as vulnerabilidades do LocalStorage, neste capítulo será feita uma demonstração prática. Durante o projeto, foi desenvolvido um sistema web que faz uso do LocalStorage, e um ataque planejado feito sobre o mesmo, de forma a exaltar a facilidade de implementação e os riscos existentes em um programa mal planejado.

## 5.1 O sistema

O sistema foi elaborado com a proposta de provar que é possível manipular os dados do HTML5 Storage de forma a burlar os métodos de segurança impostos.

O sistema é composto por duas páginas principais: o Index ou página inicial, página a qual o usuário deseja acessar, e o Login, que é utilizado para controlar o acesso ao site.

Ao acessar o sistema, o usuário é direcionado a página de login, na qual ele deve inserir os dados (email e senha) para que possa ter acesso à página principal. Se o usuário e/ou senha digitados estiverem incorretos ou não existirem, o usuário não será capaz de acessar a página principal.

A validação dos dados do usuário é feita da seguinte forma: Ao fazer o login, o navegador coleta as informações e envia ao servidor, o qual irá compará-las com as existentes no banco de dados. Se existir este usuário com esta senha, o servidor irá responder com uma chave de acesso, que será utilizada para fazer a entrada no sistema. Dessa forma, se o usuário tentar acessar a página sem ter feito o login, não existira chave de acesso, e o sistema irá bloquear o acesso e redirecionar o usuário para a página de login.

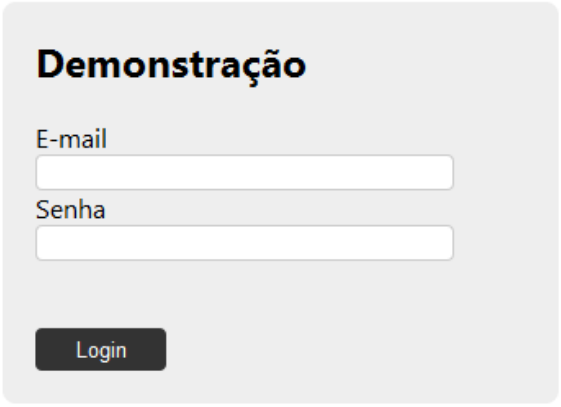



Figura 8 - Tela de login do sistema (login.html)

O sistema é composto pelos arquivos:

**- index.html**
Arquivo principal do sistema, no qual o usuário não pode acessar se não fizer login.

**- login.html**
Tela com campos para coletar credenciais do usuário e validar que este existe e está apto a acessar o arquivo principal do sistema (index.html).

**- acao.php**
Arquivo que recebe os dados do login e faz as ações necessárias para validá-los.

**- funcoes.php**
Arquivo com funções em PHP para conectar ao banco de dados, validar os dados de login e gerar a chave de acesso. Estas funções são acessadas pelo arquivo acao.php.

**- scripts.js**
Arquivo com funções em Javascript para troca de dados com o servidor.

**- estilo.css**
Folha de estilos para definir os atributos visuais do sistema.

**- banco de dados**
Sistema para armazenar usuários validos do sistema.

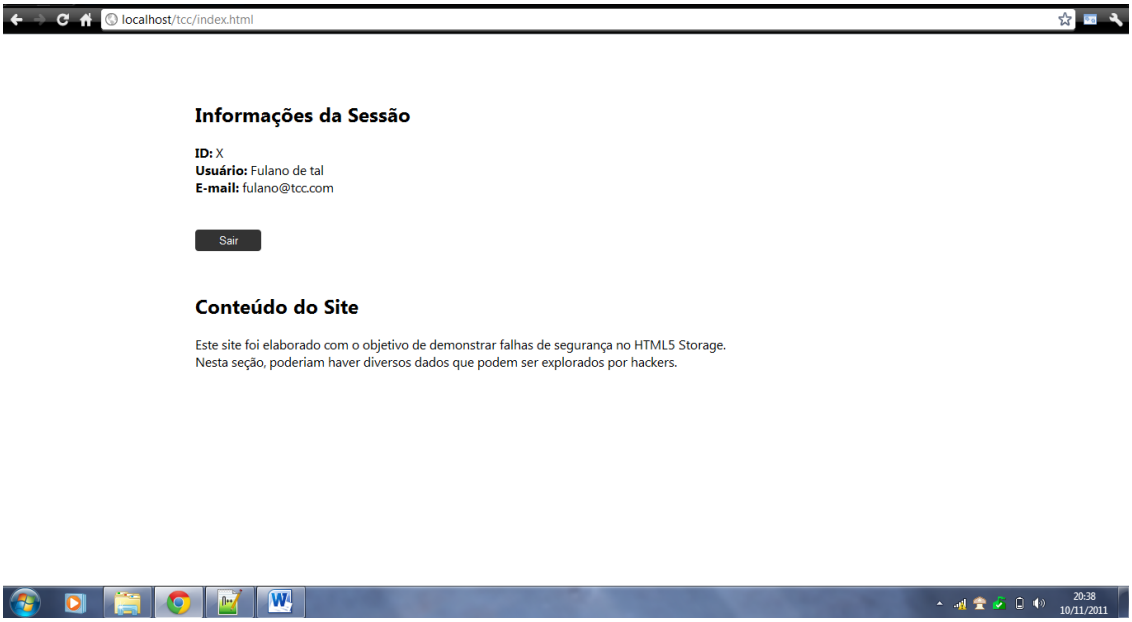



Figura 9 - Tela principal do Sistema (index.html)

## 5.2 O uso do LocalStorage no sistema

Neste sistema, o LocalStorage é utilizado para armazenar a chave de acesso enviada pelo sistema. Ao fazer um login com sucesso, o sistema armazena a chave que recebeu do servidor no LocalStorage e redireciona para a página principal.


Figura 10 - Chave armazenada no LocalStorage: acesso concedido ao sistema

Toda vez que esta última é aberta, o sistema checa se existe a chave no LocalStorage e se ela é válida. Caso sim, o acesso é concedido à página normalmente. Se não, o acesso é encerrado e o usuário é redirecionado à página de login.

## 5.3 A vulnerabilidade

Se um usuário obtiver esta chave, ele pode inseri-la manualmente no LocalStorage enquanto estiver na página de login. Com esta chave inserida, ao acessar manualmente a página index.html o usuário terá livre acesso, mesmo sem ter digitado nenhum usuário ou senha.

## 5.4 Demonstração

Primeiramente, o atacante consegue descobrir a chave de acesso através do console, enquanto um usuário está logado no sistema ou de qualquer outra forma.

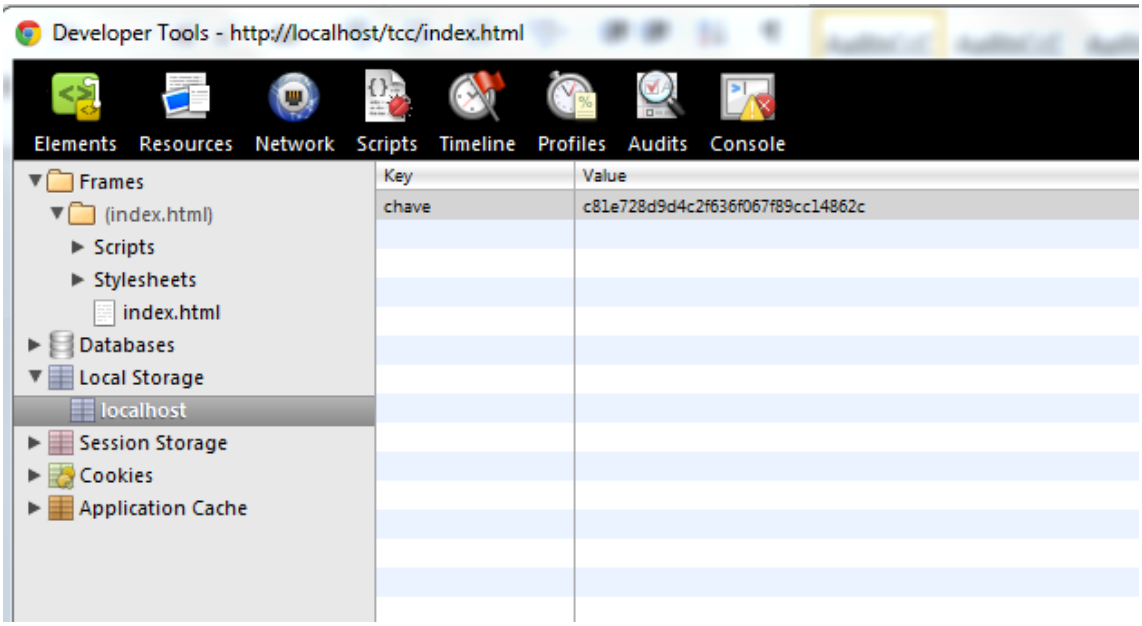



Figura 11 - Chave de acesso visualizada através das Ferramentas de Desenvolvedor

Em seguida, o atacante insere estes valores em outro navegador qualquer, enquanto estiver na tela de login do sistema ou qualquer outra página do mesmo domínio. O código utilizado para inserir os dados no LocalStorage é o mesmo em todos os navegadores. O JavaScript pode ser inserido na barra de endereços ou no console das ferramentas para desenvolvedores, caso o navegador o possua:

```
javascript:
 localStorage.setItem(“chave”, “c81e728d9d4c2f636f067f89cc14862c");
```

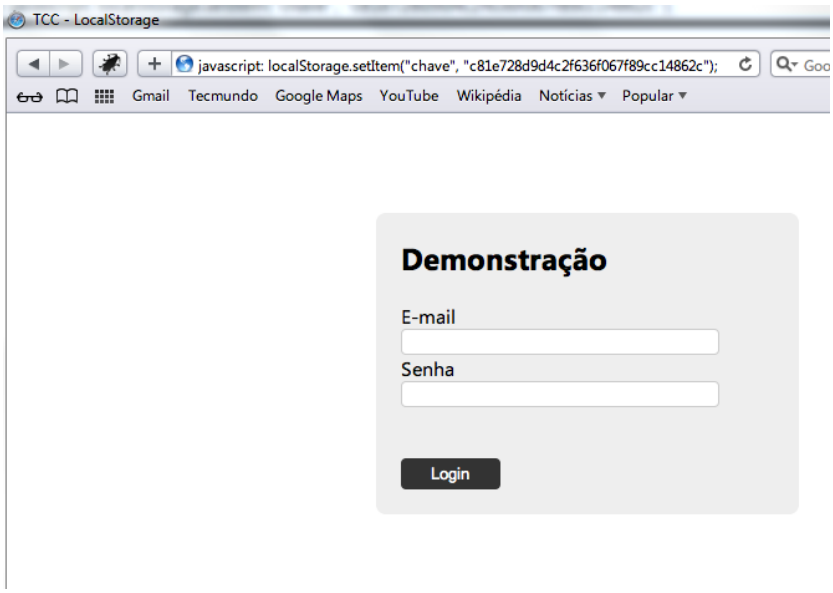



Figura 12 - Inserção da Chave de acesso via JavaScript

Finalmente, com a chave inserida manualmente no HTML5 Storage, basta acessar a página principal (index.html):

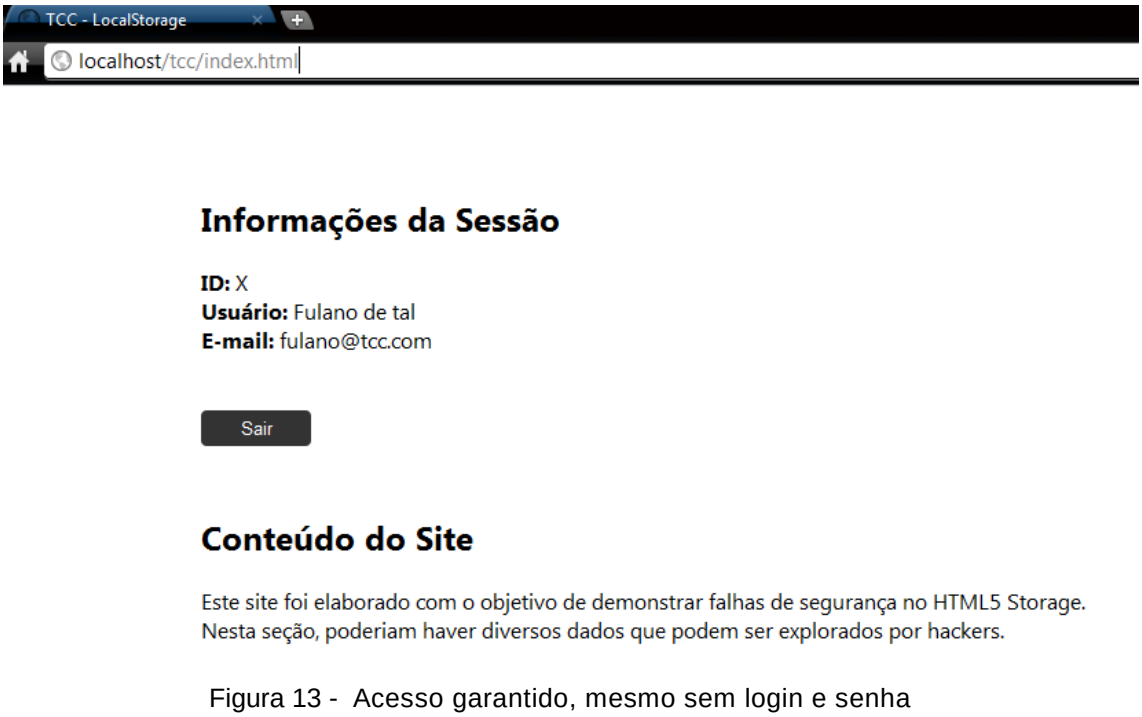



Figura 13 - Acesso garantido, mesmo sem login e senha

## 5.5 Riscos e Impactos

Essa pequena demonstração prova que apenas sabendo uma informação do usuário, no nosso caso a chave de acesso, toda a segurança do sistema pode ser facilmente sobrepujada.

Todo sistema que guarde informações no localStorage, seja do usuário ou relativa ao funcionamento do próprio sistema, dá a liberdade para que qualquer um tenha acesso a estas. Se esta informação guardar algum dado importante, qualquer atacante poderá fazer uso dela para conseguir outras informações e, talvez, entrar no sistema.

Os impactos dependem da quantidade de dados que este sistema expõe e do quão crítico é o seu funcionamento para os usuários.

# 6 CONSIDERAÇÕES FINAIS

Durante este projeto, conhecemos o HTML5 e um pouco do funcionamento da comunicação entre os usuários, os servidores, os sites e a internet como um todo. Ultimamente, cada vez mais informações estão armazenadas nos servidores e programas na nuvem. Toda a confiança deste sistema depende da capacidade e confiabilidade dos códigos produzidos pelos programadores.

O fato de que o HTML5 Storage é acessível a qualquer usuário e que este pode ser manipulado através de scripts em qualquer navegador não implica em risco imediato ao usuário, desde que as informações em questão estejam devidamente protegidas.

E há diversos meios para fazer isto. É possível criptografar todas as informações armazenadas, assim como fez a Rovio Mobile com seu jogo Angry Birds, dando fim a qualquer possibilidade de se obter acesso a níveis do qual o jogador ainda não jogou. É possível ainda criar padrões nos valores armazenados, dividindo as informações em pedaços e guardando apenas partes no cliente. Talvez ainda, o melhor jeito seja não utilizar o LocalStorage para armazenar nada que comprometa o funcionamento do sistema, buscando outros meios para guardar informações críticas do funcionamento do sistema.

Existem inúmeras maneiras de se mitigar os riscos apresentados. Só é preciso criatividade e responsabilidade dos desenvolvedores envolvidos nos projetos.

# 7 REFERÊNCIAS


Damiani, E. B. (2006). *JavaScript: Guia de Consulta Rápida.* Novatec. [1]

Dhandhania, A. (s.d.). *HTML5: Client Side Storage*. Acesso em 5 de Outubro de 2011, disponível em Web Reference: Dev the Web: http://www.webreference.com/authoring/languages/html/HTML5-Client-Side/ [2]

Foundation, M. (s.d.). *Cookies: o que são e como monitorá-los*. Acesso em 26 de Setembro de 2011, disponível em br.mozdev.org: br.mozdev.org/firefox/cookies [3]

Hickson, I. (5 de Outubro de 2011). *HTML Standard*. Acesso em 5 de Outubro de 2011, disponível em What Working Group: http://www.whatwg.org/specs/web-apps/current-work/#auto-toc-12 [4]

OpenManiak. (10 de Março de 2008). *ETTERCAP - The Easy Tutorial - Men in The Middle Attacks*. Acesso em 19 de Outubro de 2011, disponível em OpenManiak: http://openmaniak.com/ettercap_filter.php [5]

Panzarino, M. (11 de 05 de 2011). *Angry Birds for Chrome Alredy Hacked*. Acesso em 20 de Outubro de 2011, disponível em The Next Web: http://thenextweb.com/apps/2011/05/11/angry-birds-for-chrome-already-hacked-unlocking-all-levels/ [6]

Pilgrim, M. (s.d.). *Local Storage - Dive Into HTML5*. Acesso em 26 de Setembro de 2011, disponível em Dive Into HTML5: http://www.diveintohtml5.org/storage.html [7]

RegistroBR. (28 de Setembro de 2010). *Glossário, Revisão 710*. Acesso em 22 de Setembro de 2011, disponível em Registro.Br: http://registro.br/suporte/glossario.html [8]

Sanders, C. (9 de Junho de 2010). *Understanding Man In The Middle Attacks - Part2: DNS Spoofing*. Acesso em 19 de Outubro de 2011, disponível em

WindowSecurity.com: http://www.windowsecurity.com/articles/understanding-man-in-the-middle-attacks-arp-part2.html [9]

SCHWARTZ, J. (4 de Setembro de 2001). *Giving the Web a Memory Cost Its Users Privacy*. Acesso em 10 de Outubro de 2011, disponível em The New York Times: http://www.nytimes.com/2001/09/04/technology/04COOK.html [10]

Shannon, R. (08 de Abril de 2010). *Cookies | Set and Retrieve Information About Your Readers*. Acesso em 10 de Outubro de 2011, disponível em HTML Source: HTML Tutorials: http://www.yourhtmlsource.com/javascript/cookies.html [11]

*The Cookie Concept*. (s.d.). Acesso em 6 de Outubro de 2011, disponível em Cookie Central: http://www.cookiecentral.com/c_concept.htm [12]

VGFAQ. (s.d.). *Angry Birds for Chrome Cheats - VGFAQ*. Acesso em 20 de 10 de 2011, disponível em Video Games Frequently Asked Questions: http://vgfaq.com/index.php?title=Angry_Birds_for_Chrome_Cheats [13]

Vieira, L. (24 de Setembro de 2008). *ARP Poisoning*. Acesso em 19 de Outubro de 2011, disponível em iMasters: http://imasters.com.br/artigo/10117/seguranca/arp_poisoning/ [14]