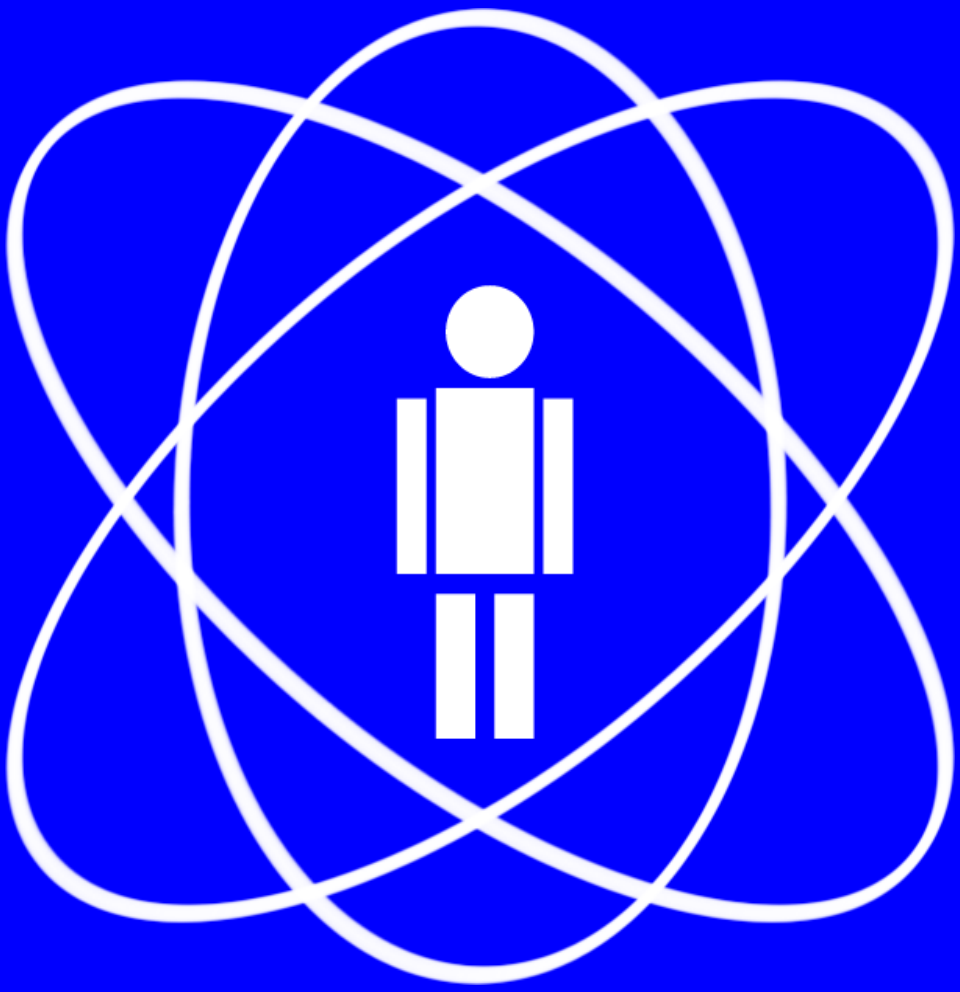


Segurança em Aplicações Web
para desenvolvedores

CAT/CBPF

Segurança em Aplicações Web *para desenvolvedores*

CAT/CBPF

25 - 27 de março de 2013

Sumário

Prefácio

Desde nosso primeiro contato com computadores que tinham comunicação virtual com outros em varias partes do mundo, na década de 80, atraves da rede BITNET - *Because It's Time NETwork*, a vida acadêmica ficou mais fácil e mais vulnerável. Mais fácil, pois temos acesso a informação com muito mais agilidade; mais vulnerável, pois podemos perder a privacidade com muito mais facilidade.

Seguindo alguns cuidados com as configurações que fazemos em nossos sistemas e aplicações, podemos ficar menos preocupados quando navegamos pela grande rede.

A intenção deste material é deixar os internautas mais atentos e ao mesmo tempo mais tranquilos. Quem aguentar verá!!

Capítulo 1

Introdução

A introdução tem por objetivo nivelar a perspectiva sobre o conteúdo usado como base para o desenvolvimento do curso. Nessa introdução veremos uma revisão sobre o protocolo HTTP e seus recursos.

1.1 Hypertext Transfer Protocol

O *Hypertext Transfer Protocol* (HTTP) ou Protocolo de Transferência de Hipertexto é um protocolo baseado em requisições e respostas usando um protocolo confiável, geralmente o protocolo TCP. [1][2]

1.1.1 Exemplo de Requisição

Acompanhe a seguir um exemplo de requisição HTTP.

```
1 GET http://www.cbpf.br/ HTTP/1.1
2 Host: www.cbpf.br
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:17.0)
    Gecko/20100101 Firefox/17.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;
    q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Proxy-Connection: keep-alive
7 Content-length: 0
```

Na primeira linha `GET http://www.cbpf.br/ HTTP/1.1` temos uma requisição do tipo GET, onde o cliente requisita a página web que gostaria de obter acesso. A requisição sempre será a primeira linha, sendo as demais cabeçalhos que servem para identificar os sistemas, tanto de origem quando de destino.

[1] RFC 2616 http://www.ietf.org/rfc/rfc2616.txt

[2] Design Considerations http://www.w3.org/Protocols/DesignIssues.html

Platform token	Descrição
Windows NT 6.2	Windows 8
Windows NT 6.1	Windows 7
Windows NT 6.0	Windows Vista
Windows NT 5.2	Windows Server 2003; Windows XP x64 Edition
Windows NT 5.1	Windows XP
Windows NT 5.01	Windows 2000, Service Pack 1 (SP1)
Windows NT 5.0	Windows 2000
Windows NT 4.0	Microsoft Windows NT 4.0
Windows 98; Win 9x 4.90	Windows Millennium Edition (Windows Me)
Windows 98	Windows 98
Windows 95	Windows 95
Windows CE	Windows CE
Macintosh; Intel Mac OS X x.y	Mac OS X em Intel x86 ou x86_64
Macintosh; PPC Mac OS X x.y	Mac OS X em PowerPC
X11; Linux i686; rv:10.0	Linux i686
X11; Linux x86_64; rv:10.0	Linux x86_64
X11; Linux i686 on x86_64; rv:10.0	Linux i686 rodando sobre x86_64

Em seguida, temos a linha `User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:17.0) Gecko/20100101 Firefox/17.0` que especifica o tipo de cliente que fez a requisição e suas capacidades. No exemplo, temos um navegador compatível com os padrões Netscape Mozilla 5.0.

Um campo *user-agent* é formado pelos seguintes subcampos ou *tokens*: **Mozilla/[version] ([system and browser information]) [platform] ([platform details]) [extensions]**. O uso comum da expressão **Mozilla** se deve que durante a *browser war*[3] muitos servidores web somente permitiam acesso a clientes compatíveis com o padrão Netscape Mozilla.

Para cada *token* existe uma gama de possíveis valores que podem variar dependendo do ambiente do cliente. Acompanhe, a seguir, uma lista dos possíveis valores para cada subcampo.

No site `http://www.useragentstring.com/pages/useragentstring.php` podemos encontrar uma lista de user agents com seus respectivos navegadores.

1.1.2 Exemplo de Resposta

Acompanhe a seguir um exemplo da resposta a requisição HTTP anteriormente analisada.

```
1 HTTP/1.1 200 OK
2 Date: Fri, 22 Mar 2013 01:31:17 GMT
3 Server: Apache
4 Accept-Ranges: bytes
```

[3] http://en.wikipedia.org/wiki/Browser_wars

```
5 Content-Length: 266
6 Content-Type: text/html
```

Na primeira linha `HTTP/1.1 200 OK` temos o status code resultante da requisição. Como podemos observar, recebemos um "200 OK" que é uma mensagem de sucesso.

Na linha 6 - `Content-Type: text/html` temos o tamanho total da resposta.

Na linha 3 - `Date: Fri, 22 Mar 2013 01:31:17 GMT` temos a data e horário que a resposta foi realizada. Isso é especialmente útil quando vamos realizar uma análise correlacionando eventos.

Opcionalmente nessa resposta, podemos observar o campo `Server: Apache` que indica o sistema pelo qual a resposta se originou.

1.1.3 URI

URI ou *Uniform Resource Indicators* é o endereço e um recurso incluindo a forma de acesso a ele. Atualmente (2013) é mais comum o uso de URL *Uniform Resource Locator* com esse significado.

O formato de uma URI é:

`protocol://[user:password@]host.domainname[:port]/resource?param=value`

A seguir, acompanhe um exemplo de URI e sua formação:

`http://portal.cbpf.br/index.php?page=home`

http -> protocol

portal -> host

cbpf.br ->domain name

index.php -> resource

page -> param

home -> value

Ao manipularmos os elementos da URI estamos modificando como as requisições GET serão realizadas para o servidor web. Geralmente, uma aplicação web comum passa seus parâmetros seguindo o seguinte padrão: `?param=value&param2=value2`.

Existem outras formas de passar esses mesmo dados para o servidor web, sem realizar o uso dos caracteres "?", "&" e "=", que são muito comuns. Para isso deve ser feito o uso de recursos no servidor web para tratar as requisições GET e repassá-las de forma adequada a aplicação web. Como exemplo, temos o *mod_ rewrite* presente no servidor web Apache que faz essa função, assim a URI vista anteriormente ficaria da forma: `/param/value/param/value2`.

1.1.4 Métodos

Os clientes de um servidor web podem usar vários métodos disponíveis para realizar suas requisições a um servidor web. Ao todo existem 8 métodos disponíveis, sendo eles: GET, POST, HEAD, TRACE, OPTIONS, CONNECT, PUT, DELETE.

Os métodos mais importantes são: GET e POST.

O método **GET** serve para solicitar páginas ao servidor web, podendo também enviar dados como parâmetros.

O método **POST** serve para enviar dados ao servidor web para processamento. A diferença principal entre os métodos é que o método GET envia os dados via URI e o método POST envia os dados usando o *payload* HTTP.

Os outros métodos servem respectivamente para:
HEAD - Obtém somente o cabeçalho HTTP. É uma variação do GET sem obter os recursos solicitados.

Acompanhe o exemplo a seguir(requisição e resposta).

```
1 HEAD http://portal.cbpf.br HTTP/1.1
2 Host: portal.cbpf.br
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:17.0)
    Gecko/20100101 Firefox/17.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;
    q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Proxy-Connection: keep-alive
7 Content-length: 0
```

```
1 HTTP/1.1 200 OK
2 Date: Fri, 22 Mar 2013 02:28:50 GMT
3 Server: Apache
4 X-Powered-By: PHP/5.2.8
5 Set-Cookie: CBPF=224r75rcj3aumndbiv5ok1c5n3; path=/
6 Expires: Thu, 19 Nov 1981 08:52:00 GMT
7 Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
    pre-check=0
8 Pragma: no-cache
9 Vary: Accept-Encoding
10 Content-Type: text/html;charset=iso-8859-1
```

PUT - Envia um recurso

Acompanhe o exemplo a seguir.

```
1 PUT http://portal.cbpf.br HTTP/1.1
2 Host: portal.cbpf.br
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:17.0)
    Gecko/20100101 Firefox/17.0
```

```
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;
       q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Proxy-Connection: keep-alive
7  Content-length: 0
```

```
1  HTTP/1.1 200 OK
2  Date: Fri, 22 Mar 2013 02:28:50 GMT
3  Server: Apache
4  X-Powered-By: PHP/5.2.8
5  Set-Cookie: CBPF=lot9g380qqkhe08c69spcdtj84; path=/
6  Expires: Thu, 19 Nov 1981 08:52:00 GMT
7  Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
       pre-check=0
8  Pragma: no-cache
9  Vary: Accept-Encoding
10 Content-Type: text/html;charset=iso-8859-1
```

DELETE - Exclui um recurso
Acompanhe o exemplo a seguir.

```
1  DELETE http://portal.cbpf.br HTTP/1.1
2  Host: portal.cbpf.br
3  User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:17.0)
       Gecko/20100101 Firefox/17.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;
       q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Proxy-Connection: keep-alive
7  Content-length: 0
```

```
1  HTTP/1.1 200 OK
2  Date: Fri, 22 Mar 2013 02:28:50 GMT
3  Server: Apache
4  X-Powered-By: PHP/5.2.8
5  Set-Cookie: CBPF=1r8h5mvgp2be72u41s1ak04b33; path=/
6  Expires: Thu, 19 Nov 1981 08:52:00 GMT
7  Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
       pre-check=0
8  Pragma: no-cache
9  Vary: Accept-Encoding
10 Content-Type: text/html;charset=iso-8859-1
```

TRACE - Ecoa o pedido, de maneira que o cliente possa saber o que os servidores intermediários estão mudando em seu pedido.

Acompanhe o exemplo a seguir.

```
1 TRACE http://portal.cbpf.br HTTP/1.1
2 Host: portal.cbpf.br
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:17.0)
    Gecko/20100101 Firefox/17.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;
    q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Proxy-Connection: keep-alive
7 Content-length: 0
```

```
1 HTTP/1.1 403 Forbidden
2 Date: Fri, 22 Mar 2013 02:28:50 GMT
3 Server: Apache
4 Vary: Accept-Encoding
5 Content-Length: 202
6 Content-Type: text/html; charset=iso-8859-1
```

OPTIONS - Verifica os métodos permitidos pelo servidor web.
Acompanhe o exemplo a seguir.

```
1 OPTIONS http://portal.cbpf.br HTTP/1.1
2 Host: portal.cbpf.br
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:17.0)
    Gecko/20100101 Firefox/17.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;
    q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Proxy-Connection: keep-alive
7 Content-length: 0
```

```
1 HTTP/1.1 200 OK
2 Date: Fri, 22 Mar 2013 02:28:50 GMT
3 Server: Apache
4 X-Powered-By: PHP/5.2.8
5 Set-Cookie: CBPF=sfkeukpas8mte3nv57m858a5f2; path=/
6 Expires: Thu, 19 Nov 1981 08:52:00 GMT
7 Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
    pre-check=0
8 Pragma: no-cache
9 Vary: Accept-Encoding
10 Content-Type: text/html;charset=iso-8859-1
```

CONNECT - Abre um túnel para o servidor ser usado como proxy
Acompanhe o exemplo a seguir.


```
1 CONNECT http://portal.cbpf.br HTTP/1.1
2 Host: portal.cbpf.br
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:17.0)
    Gecko/20100101 Firefox/17.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;
    q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Content-length: 0
```

```
1 HTTP/1.1 400 Bad Request
2 Date: Fri, 22 Mar 2013 02:46:45 GMT
3 Server: Apache
4 Vary: Accept-Encoding
5 Content-Length: 226
6 Connection: close
7 Content-Type: text/html; charset=iso-8859-1
```

1.1.5 Códigos de Resposta

O protocolo HTTP usa códigos numéricos para identificar o tipo de resposta. Existem 5 classes, com as seguintes designações.

1xx Informational

- 100 Continue

2xx Success

- 200 OK

3xx Redirection

- 301 Moved Permanently
- 302 Redirect
- 304 Not Modified

4xx Client Error

- 400 Bad Request
- 401 Not Unauthorized
- 403 Forbidden
- 404 File not found

5xx Server Error

- 500 Server Error
- 502 Bad Gateway

Uma lista mais extensa dos Códigos de Resposta pode ser encontrada em: `http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html`.

1.1.6 Cookies

Os *cookies* são elementos do protocolo HTTP que são enviados ao navegador pelo servidor, com o objetivo de lembrar informações de um usuário. É formado por uma cadeia de caracteres, organizados de forma cadeia/valor, separados por ponto-e-vírgula. Seus principais usos são a manutenção de uma sessão (que não e suportado pelo protocolo) e a autenticação de usuários.

A seguir, podemos observar essas características no cookie enviado pelo www.cbpf.br.

```
Set-Cookie:  CBPF=sfkeukpas8mte3nv57m858a5f2; path=/
```

Alguns atributos de um *cookie* são:

expires - define por quanto tempo o cookie é valido

domain - define para quais domínios o cookie é valido

path - define os caminhos para os quais o cookie é válido

secure - obriga o cookie a ser enviado somente em requisições via HTTPS

HttpOnly - impede que o cookie seja acessado por código executado no lado do cliente

1.1.7 Autenticação HTTP

A autenticação serve para identificar o usuário para a aplicação web ou para o servidor web. Geralmente é usada uma combinação de usuário e senha, o que garante somente um dos requisitos para uma boa autenticação.

Os requisitos para uma boa autenticação se baseiam em:

Algo que você sabe - geralmente uma senha

Algo que você é - geralmente um ser humano (biometria)

Algo que você tem - geralmente um token ou smartcard

O protocolo HTTP possui dois métodos nativos para autenticação: *Basic* e *Digest*. Embora existam somente esses dois métodos no protocolo, a autenticação geralmente pode ser feita utilizando certificados no cliente, de forma integrada ao ambiente de rede (*Single Sign On* [4]), ou baseada em formulários.

[4] `http://en.wikipedia.org/wiki/Single_sign-on`

Autenticação Basic

A autenticação usando o método Basic[5] é bem simples, não utilizando nem criptografia. Seu conteúdo é transmitido utilizando ofuscação, com a codificação dos dados em base64[6].

Não existe nenhum controle de força bruta, isso possibilita,por exemplo, encher o disco com os logs do servidor web.

Autenticação Digest

A autenticação usando o método Digest[7] é bem simples também, porém utiliza MD5 que é um hash criptográfico já em desuso devido sua fragilidade.

Como o MD5 faz apenas uma passagem sobre os dados, se dois prefixos com o mesmo hash forem construídos, um sufixo comum pode ser adicionado a ambos para tornar uma colisão mais provável. Deste modo é possível que duas strings diferentes produzam o mesmo hash. O que não garante que a partir de uma senha criptografada específica consiga-se a senha original, mas permite uma possibilidade de decifrar algumas senhas a partir de um conjunto grande de senhas criptografadas.

Autenticação baseada em formulário

É a forma mais comum de autenticação atualmente, tanto pelo controle quanto pela facilidade de uso.

É recomendado que os programadores façam uso da API da OWASP[8] devido sua maturidade e sua preocupação com a segurança em diversas camadas.

1.1.8 Controle de Sessão

O protocolo HTTP é um protocolo sem controle de estado, logo a aplicação deve implementar uma forma de fazer o agrupamento de requisições em uma sessão, ou seja, a aplicação deve fazer o controle de sessão.

Os métodos mais usados para fazer o controle de sessão são através de:

- cookies
- codificação da URL
- campos hidden em formulários

[5]RFC 2617
[6]`http://pt.wikipedia.org/wiki/Base64`
[7]RFC 2617
[8]https://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API

Tipos de Sessão

Existem dois tipos de sessão: *client-side* e *server-side*.

No tipo *client-side* todos os dados relativos a sessão são enviados para o cliente, o que é muito conveniente para um indivíduo mal intencionado, pois a aplicação acredita que o cliente não modificará os dados enviados.

No tipo *server-side*, a sessão é mantida no servidor e é controlada através do uso de uma variável, conhecida como session ID. Nesse último caso, somente uma variável pode ser manipulada, a session ID.

Cookies

Como vimos anteriormente os *cookies* são elementos do protocolo HTTP que são enviados ao navegador pelo servidor, com o objetivo de lembrar informações de um usuário. Eles possuem alguns recursos de segurança, de modo a permitir, por exemplo, que o cookie só possa ser manipulado via HTTPS. Também podem receber marcações para permitir seu armazenamento somente em memória ou ter validade por um determinado tempo.[9]

Codificação de URL

Geralmente uma URI somente pode conter caracteres ASCII[10] , sendo que alguns deles apresentam funções especiais, como delimitadores ou com uso reservado. Para utilizá-los como dados para a aplicação web é necessário codificar esses caracteres.

A seguir, veja uma lista dos caracteres reservados:

```
":"
"/"
"?"
"#"
"["
"]"
"@"
"!"
"$"
"&"
"'"
"("
")"
"*"
"+"
","
";"
```

[9]RFC 2965
[10]RFC 3986

"="

O método padrão de codificação de caracteres consiste no uso de um caractere "%" seguido da representação em hexadecimal do caractere.

Confira em `http://www.w3schools.com/tags/ref_urlencode.asp` uma tabela com a codificação de diversos caracteres.

A codificação é realizada através da chave: **parâmetro=valor** de modo que quando um usuário clica em determinado link, o navegador envia os dados via GET para o servidor web.

Veja o próximo exemplo, onde o parâmetro `id` recebe o valor **580**. Mudando o valor para **579**, por exemplo, encontramos outra página, com outro contexto.

- `http://portal.cbpf.br/index.php?page=Noticias.VerNoticia&id=580` -> Notícia sobre colóquio
- `http://portal.cbpf.br/index.php?page=Noticias.VerNoticia&id=579` -> Notícia sobre prorrogação de prazo

Campos Hidden

Nesse caso, a aplicação retorna ao usuário formulários com campos do tipo *hidden* que não aparecem para os usuários. Não somente, mas os fomulários podem conter um mistura de campos visíveis com campos *hidden*.

No exemplo encontrado no site: `http://www.rederio.br/site/contact` encontramos um campo do tipo *hidden* responsável por fazer o controle do preenchimento do formulário. Veja-o a seguir.

```
<input name="captcha_token"value="91051e5ff489d1e5a06e3fdcd043eba0"/>
```

Em outros casos, após o usuário estar autenticado em uma aplicação, é possível encontrar formulários com um campo com a identificação do usuário, como, por exemplo, um campo "id". Nesse tipo de campo, podemos tentar subverter o sistema de controle de sessão, alterando assim seu valor para tentar obter acesso a conta de outro usuário.

1.1.9 HTTPS

O protocolo HTTPS consiste na implementação de um canal SSL/TLS para o transporte de dados HTTP. Com o uso do canal SSL/TLS a comunicação passa a trafegar de forma encriptada. Sua segurança está baseada no conceito de Web of Trust [11] segundo a SANS, mas da forma que conhecemos, ela está mais para o conceito hierárquico de PKI [12].

O conceito de PKI está baseado na confiança de uma *Autoridade Raiz* que certifica *Autoridades Intermediárias* através da geração de certificados digitais. O comprometimento de uma Autoridade Certificadora possui consequências enormes pois a revogação de seus

[11] http://en.wikipedia.org/wiki/Web_of_trust
[12] http://en.wikipedia.org/wiki/Public_key_infrastructure

certificados pode ser muito demorada para evitar incidentes graves e muito custosa para seus clientes.

Veja, a seguir, algumas notícias sobre um comprometimento que ocorreu em 2011.

- `http://www.gris.dcc.ufrj.br/news/apple-lanca-atualizacao-de-seguranca-para-certific`
- `http://www.computerworld.com/s/article/9219663/Hackers_may_have_stolen_over_200_SSL_certificates`
- `http://technet.microsoft.com/en-us/security/advisory/2607712`

Problemas do HTTPS

Além do risco do comprometimento de uma autoridade certificadora, um outro problema do uso de HTTP é o uso cada vez mais frequentes de certificados auto-assinados. O uso do certificado auto-assinado não é um problema, mas as informações que são passadas aos usuários para contornar as mensagens de erro apresentados por seu uso são.

Veja na imagem 1.1 o exemplo de instruções para contornar o uso de certificados auto-assinados.

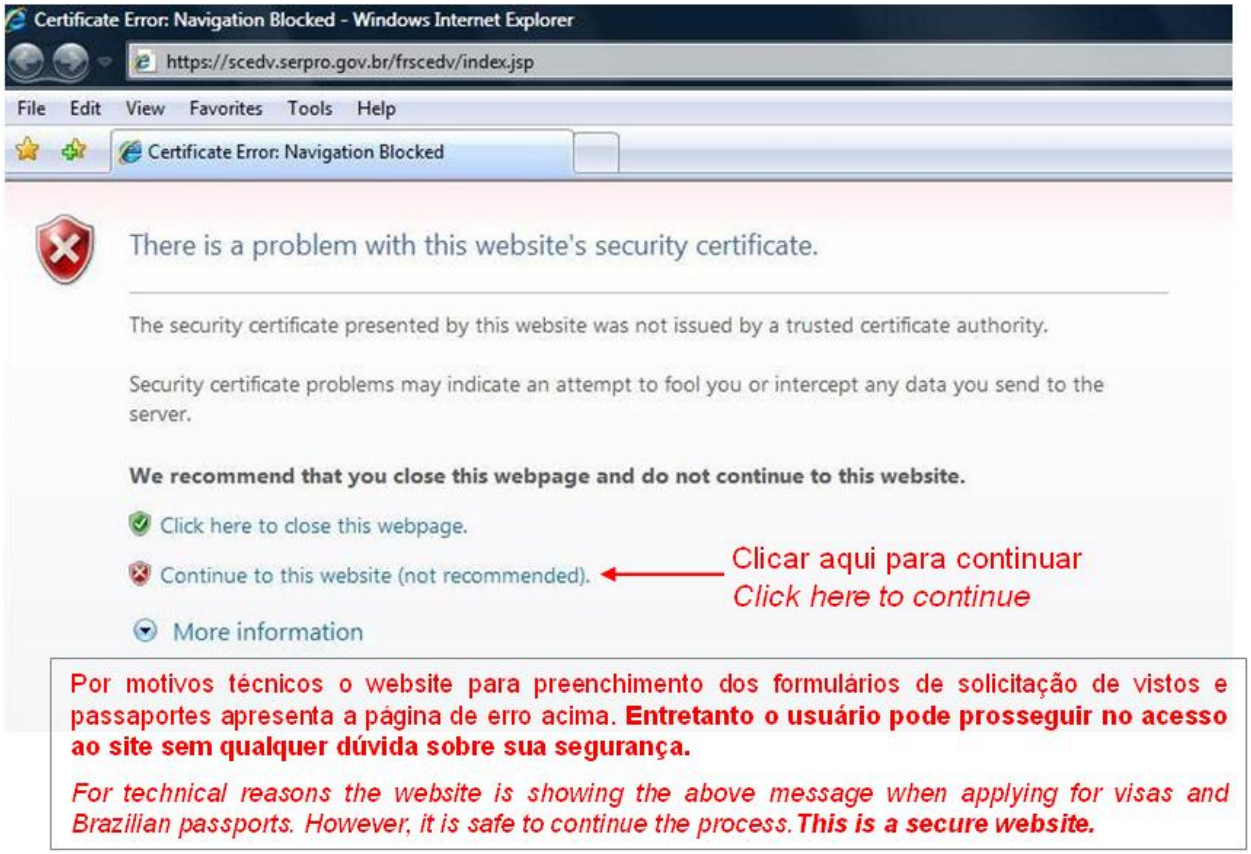


Figura 1.1: Instruções de uso para certificados auto-assinados - Fonte: `http://chicago.itamaraty.gov.br/pt-br/maiores_18_anos_-_consulado.xml`

Capítulo 2

Auditorias Web

> Se você conhece o inimigo e conhece a si mesmo, não precisa temer o resultado de cem batalhas. Se você se conhece mas não conhece o inimigo, para cada vitória ganha sofrerá também uma derrota. Se você não conhece nem o inimigo nem a si mesmo, perderá todas as batalhas...
>
> *Sun Tzu*

2.1 Segurança das Aplicações *versus* Segurança da Comunicação

Diversos desenvolvedores de soluções web acreditam estar protegendo suas aplicações com através da implantação de certificados nos seus servidores. Existe no mercado, até empresas que vendem sistemas que monitoram e testam a qualidade dos certificados SSL dizendo oferecer aos seus clientes uma melhor segurança em suas aplicações web. Na realidade, o uso de certificados SSL somente atua na segurança da comunicação, garantindo assim a confidencialidade dos dados trafegados.

Em nenhum momento o SSL protege a aplicação de falhas, até porque o SSL não é implantado na aplicação, e sim no serviço que disponibilizará a aplicação.

2.2 Aplicações Web mais comuns

O desenvolvimento de aplicações web gerenalistas tem aumentado devido a grande demanda de aplicações que tivessem um núcleo robusto juntamente com uma alta capacidade de customização.

Para isso, aplicações web começaram a ser desenvolvidas como plataformas de conteúdo e sua customização começou a ser feita com a instalação de módulos.

Aplicações Web comuns são:

- CMS

- Wikis
- Lojas Virtuais

2.3 Tipos de Auditorias

2.3.1 BlackBox vs WhiteBox

Teste **caixa-preta** ou Teste **BlackBox** é um teste de software para verificar a saída dos dados usando entradas de vários tipos. Tais entradas, geralmente, não são escolhidas conforme a estrutura do programa.

Quanto mais entradas são fornecidas, mais rico será o teste. Numa situação ideal todas as entradas possíveis seriam testadas, mas na ampla maioria dos casos isso é impossível.

O teste **caixa preta** é um teste em que a principal característica é que seu "testador" não possui conhecimento prévio sobre o funcionamento da aplicação.

Teste de **caixa-branca** ou teste **WhiteBox** é uma técnica de teste que usa a perspectiva interna do sistema para modelar os casos de teste. No teste de software, a perspectiva interna significa basicamente o código-fonte.

A principal diferença do teste **caixa-preta** para o teste **caixa-branca** é que a perspectiva interna do sistema é desconsiderada, sendo testadas e mensuradas somente as interfaces do sistema. Entretanto, ambas as técnicas podem ser usadas em conjunto, no que é chamado teste **caixa-cinza**. Dessa forma, o teste é modelado conhecendo-se a estrutura interna do sistema, mas a execução ignora esse aspecto, como no teste **caixa-preta**.

2.4 Metodologia de ataque

A metodologia de ataque à uma aplicação web é cíclica. Ele consiste de 4 fases[1], sendo eles:

Reconhecimento - Fase de pesquisa sobre a aplicação

Mapeamento - Fase para entender como a aplicação funciona

Descoberta - Fase de busca por vulnerabilidades

Exploração - Fase de execução dos ataques

São fases iterativas e realimentadas, pois a cada fase temos uma nova perspectiva sobre a aplicação.

[1] De acordo com a metodologia da SANS, uma das principais referencias no assunto.

2.4.1 Reconhecimento

A fase da pesquisa sobre a aplicação é importante para determinar o foco dos ataques. Nessa fase é interessante determinar os diagramas de arquitetura da aplicação, informações sobre o procedimento de autenticação, informações sobre o procedimento de recuperação de senha, informações sobre a infraestrutura, como caches, balanceadores de carga, firewall de aplicação, entre outras informações sobre a rede alvo.

2.4.2 Mapeamento

A fase para entender como a aplicação funciona em conjunto com sua infraestrutura. Os primeiros passos envolvem a execução de varreduras de portas, checagem de versão de aplicações e dependências, assim como a determinação do sistema operacional e do servidor web.

Nessa fase, também deverão ser mapeados os relacionamentos da aplicação com outros conteúdos, como, por exemplo, outras páginas dentro do mesmo servidor web. Diversas aplicações contam com recursos como o *sitemap* ou mapa do site, que fornece um bom começo para essa fase.

2.4.3 Descoberta

A fase de busca por vulnerabilidade é importante para correlacionar os resultados no mapeamento com as possíveis ameaças que a aplicação web está exposta. Nessa fase, o foco estará na procura de informações sobre o desenvolvimento da aplicação, informações sobre os sistemas e arquivos da aplicação, sistemas de autenticação, API's, mensagens de erros e possíveis problemas no uso da aplicação.

Algumas informações são importantes nessa fase. como, por exemplo, nomes de usuários, possíveis ID's, possíveis valores para parâmetros, etc. Nessa fase, também vamos determinar que ferramentas são interessantes para a fase de exploração.

2.4.4 Exploração

Na fase da execução dos ataques é importante lembrar de documentar todos os passos dados e seus resultados. Durante a execução dos ataques, cada ataque executado com sucesso é uma interface para descobrir mais informações sobre a aplicação. Com isso, podem ser descobertas novas partes da aplicação, outras aplicações que compartilham recursos ou que podem ser acessadas através dela.

2.5 Tipos de falhas

A seguir veremos algumas categorias de falhas esperadas em aplicações web. Essa listagem não é extensiva é somente para termos uma noção antes de entrarmos mais profundamente nos próximos capítulos.

2.5.1 Exposição de Informações

Esse tipo de falha permite descobrir informações sobre a aplicação. Esse tipo de informação pode ser: informações sobre a infraestrutura, informações sobre correções, informações sobre versão de dependências externas, informações sobre armazenamento de dados e informações sobre usuários e senhas.

2.5.2 Falhas de Configuração e Manutenção

Esse tipo de falha é comum em servidores web que não foram configurados de forma específica para a aplicação. Assim, esses servidores web, com sua configuração padrão expõem sua assinatura, versão, e outras informações que podem ser úteis para ataques direcionados.

2.5.3 Falhas de Validação de Dados

Vulnerabilidades relacionadas com a Validação de Dados costumam ser as mais presentes em aplicações, mesmo em aplicações desktop.

Embora sejam as mais comuns, elas representam um grande risco para a aplicação. Primeiro, porque permitem a manipulação da informação presente na aplicação e segundo, porque podem facilmente denegrir a imagem da organização dona da aplicação.

2.5.4 Falhas de Disponibilidade

Categoriza vulnerabilidades que afetam a capacidade do sistema em responder às requisições feitas pelos clientes (usuários, outros sistemas, monitoramento, ...) conforme as regras que são estabelecidas para cada sistema. Ou seja, afetam a capacidade do serviço estar disponível quando requisitado.

Capítulo 3

Reconhecimento

Essa é a primeira fase das quatro existentes na metodologia de teste de invasão em aplicações web[1].

Existem diversas maneiras de iniciar o reconhecimento, entre elas pesquisas na Internet, pesquisas direcionadas com o GHDB - Google Hacking Database [2], pesquisas em forums, mídias sociais e blogs pessoais.

- Informações que podem ser encontradas:
- Nomes de funcionários
- Identificadores de usuários
- Informações diversas sobre usuários
- Tecnologias empregadas
- Servidores e Topologia de Rede
- Configuração dos componentes
- Recursos disponibilizados pelos servidores web
- Arquivos "robots.txt"

Outras informações importantes são relacionadas a infraestrutura. Exemplos são:

- Balanceadores de Carga
- Firewall de aplicação Web
- Proxies
- Informações sobre o host, como vulnerabilidades e versões de sistema

[1] SANS

[2] http://www.hackersforcharity.org/ghdb/

3.1 WHOIS

O *whois*[3] é um protocolo usado com a finalidade de prover um serviço de identificação para blocos de endereços IP.

Antes de realizar consultas *whois* é interessante estipular sobre qual entidade de registro o domínio ou bloco de IPs foi registrado. Internacionalmente, existem entidades para cada continente ou parte do continente. A seguir, se encontra uma lista com as entidades e na imagem 3.1, um mapa com a área de atuação de cada uma.

- AFRINIC (África) - http://www.afrinic.net
- APNIC (Ásia e Oceania) - http://www.apnic.net
- ARIN (EUA, Canadá e Antártida) - http://ws.arin.net
- LACNIC (América Latina) - http://www.lacnic.net
- RIPE (Europa, Rússia e Oriente Médio) - http://www.ripe.net

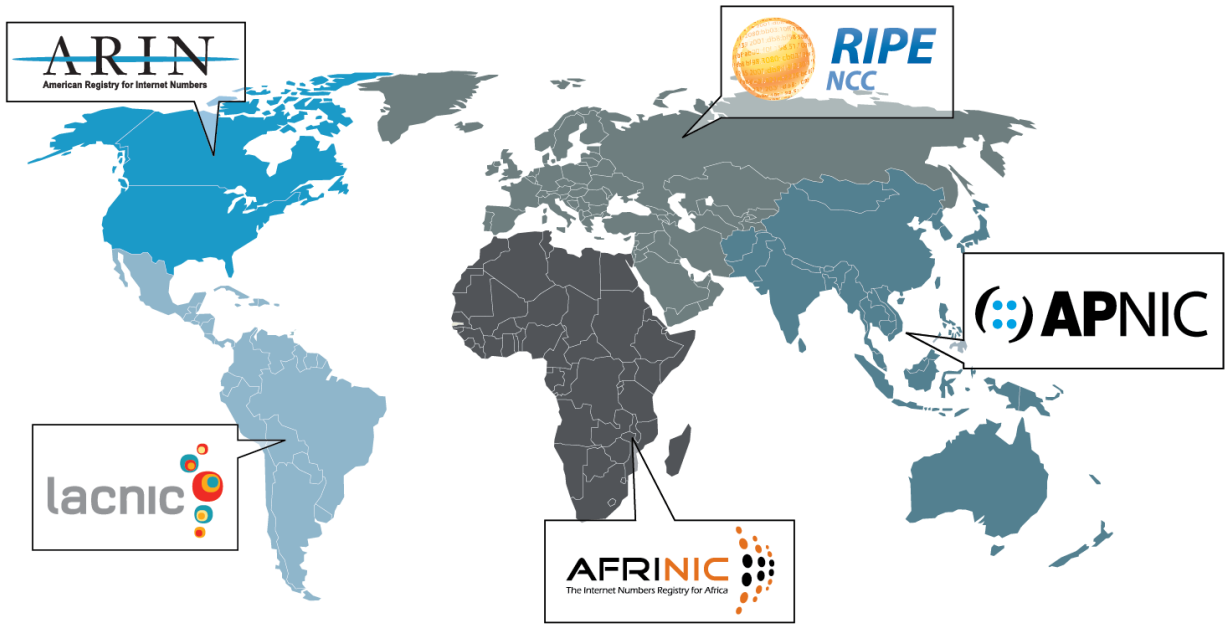


Figura 3.1: Entidades em cada continente

Acompanhe alguns exemplos de uso do *whois* a seguir:

[3]RFC 3912 - `http://tools.ietf.org/html/rfc3912`

```
$ whois -h whois.registro.br cbpf.br

% Copyright (c) Nic.br
%  The use of the data below is only permitted as described in
%  full by the terms of use (http://registro.br/termo/en.html),
%  being prohibited its distribution, comercialization or
%  reproduction, in particular, to use it for advertising or
%  any similar purpose.
%  2013-03-22 08:54:06 (BRT -03:00)

domain:      cbpf.br
owner:       CENTRO BRASILEIRO DE PESQUISAS FISICAS
ownerid:     033.654.831/0014-50
responsible: Marita Maestrelli
country:     BR
owner-c:     MAM214
admin-c:     MAM214
tech-c:      MAM214
billing-c:   MAM214
nserver:     cbpfsu1.cat.cbpf.br 152.84.253.2
nsstat:      20130320 AA
nslastaa:    20130320
nserver:     p0m1.cat.cbpf.br 152.84.253.220
nsstat:      20130320 AA
nslastaa:    20130320
nserver:     cbpfsu2.cat.cbpf.br 152.84.253.3
nsstat:      20130320 AA
nslastaa:    20130320
dsrecord:    22822 RSA/SHA-1 8D87F19358A2CD935A5FDD94016326CA42062764
dsstatus:    20130320 DSOK
dslastok:    20130320
dsrecord:    22822 RSA/SHA-1 299970DF51DDB96A5AC811B015C26CC981D44EBB32C6216CA1525153A31
dsstatus:    20130320 DSOK
dslastok:    20130320
created:     before 19950101
changed:     20121206
status:      published

nic-hdl-br:  MAM214
person:      Marita Maestrelli
e-mail:      marita@cbpf.br
created:     19980904
```

```
changed:     20030811

% Security and mail abuse issues should also be addressed to
% cert.br, http://www.cert.br/, respectivelly to cert@cert.br
% and mail-abuse@cert.br
%
% whois.registro.br accepts only direct match queries. Types
% of queries are: domain (.br), ticket, provider, ID, CIDR
% block, IP and ASN.
```

Podemos fazer algumas consultas para obter mais informações sobre domínios, acompanhe a seguir:

```
$ whois -h whois.registro.br 033.654.831/0014-50

% Copyright (c) Nic.br
%  The use of the data below is only permitted as described in
%  full by the terms of use (http://registro.br/termo/en.html),
%  being prohibited its distribution, comercialization or
%  reproduction, in particular, to use it for advertising or
%  any similar purpose.
%  2013-03-22 08:57:33 (BRT -03:00)

owner:       CENTRO BRASILEIRO DE PESQUISAS FISICAS
ownerid:     033.654.831/0014-50
responsible: Marita Maestrelli
country:     BR
owner-c:     MAM214
created:     19980904
changed:     20081007

nic-hdl-br:  MAM214
person:      Marita Maestrelli
e-mail:      marita@cbpf.br
created:     19980904
changed:     20030811

domain:      cbpf.br

% Security and mail abuse issues should also be addressed to
% cert.br, http://www.cert.br/, respectivelly to cert@cert.br
% and mail-abuse@cert.br
%
```



```
% whois.registro.br accepts only direct match queries. Types
% of queries are: domain (.br), ticket, provider, ID, CIDR
% block, IP and ASN.
```

Existem outras ferramentas mais dinâmicas ou que fornecem mais informações. Alguns exemplos são:

- http://www.team-cymru.org/Services/ip-to-asn.html (recomendado para scripts que fazem muitas consultas)
- http://www.geektools.com/whois.php

3.2 DNS

O sistema de nomes de domínio, do inglês *Domain Name System* - DNS é um sistema de gerenciamentos de nomes que tem por finalidade permitir a identificação dos hosts na internet. Ou seja, o sistema de DNS traduz nomes de domínios em endereços IP e vice-versa [4].

3.2.1 Exemplos com dig e nslookup

Vejamos a seguir um exemplo de uma consulta usando o nslookup:

```
$ nslookup www.cbpf.br
Server: 187.100.246.251
Address: 187.100.246.251#53

Non-authoritative answer:
Name: www.cbpf.br
Address: 152.84.253.9
```

Podemos realizar a mesma consulta utilizando a ferramenta dig que possui funcionalidades semelhantes, que enumeraremos mais adiante:

```
$ dig www.cbpf.br

; <<>> DiG 9.8.3-P1 <<>> www.cbpf.br
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 25850
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
```

[4] RFC 1034 e RFC 1035

Record	Tipo	Uso
A	Registro de hosts	Associa um nome a um endereço IP
MX	Registro de servidores de email	Associa um endereço IP ao serviço de correio
PTR	Registro de ponteiro	Associa um endereço IP a um nome
NS	Registro de servidores de nome	Associa um endereço IP ao serviço de DNS
SOA	Registro de autoridade da zona	Registro dos servidores autoritativos da zona
NS	Registro de servidores de nome	Associa um endereço IP ao serviço de DNS
SOA	Registro de autoridade da zona	Registro dos servidores autoritativos da zona

Tabela 3.1: Tipos e Usos de registros DNS

```
;; QUESTION SECTION:
;www.cbpf.br. IN A

;; ANSWER SECTION:
www.cbpf.br. 600 IN A 152.84.253.9

;; Query time: 201 msec
;; SERVER: 187.100.246.251#53(187.100.246.251)
;; WHEN: Fri Mar 22 09:02:11 2013
;; MSG SIZE  rcvd: 45
```

3.2.2 Tipos de registro

Outra característica do sistema de nomes é que cada registro, além de um endereço IP e um nome, também possui um tipo. O tipo é chamado internamente pelo sistema de *resource records* e cada tipo de registro serve para armazenar um tipo de informação.

Acompanhe na tabela 3.1 os tipos e usos para cada um.

Para realizar consultas por tipo usando a ferramenta *dig* faça conforme o exemplo a seguir:

```
$ dig -t SOA www.cbpf.br

; <<>> DiG 9.8.3-P1 <<>> -t SOA www.cbpf.br
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31025
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;www.cbpf.br. IN SOA

;; AUTHORITY SECTION:
```



```
cbpf.br. 600 IN SOA cbpfsu1.cat.cbpf.br. marita.cbpfsu1.cat.cbpf.br. 2013031900 3600 900

;; Query time: 4810 msec
;; SERVER: 200.142.130.202#53(200.142.130.202)
;; WHEN: Fri Mar 22 09:10:16 2013
;; MSG SIZE  rcvd: 84
```

3.3 Newsgroups e Anúncios de Empregos

A participação de funcionários de organizações em *newsgroups* e listas de discussão é bem conhecida principalmente de funcionários dos departamentos relacionados a Tecnologia da Informação.

Através do uso de seus nomes ou até de seus emails corporativos (que podem ser obtidos nas fases anteriores), é possível encontrar mensagens que podem revelar suas preferências de uso, preferências por softwares específicos ou, em casos mais graves, pedidos de ajuda na configuração de serviços, incluindo a exposição de arquivos de configuração dos sistemas da empresa.

Acompanhe o exemplo na imagem 3.2, onde se encontra um pedido de ajuda na configuração de um determinado serviço.

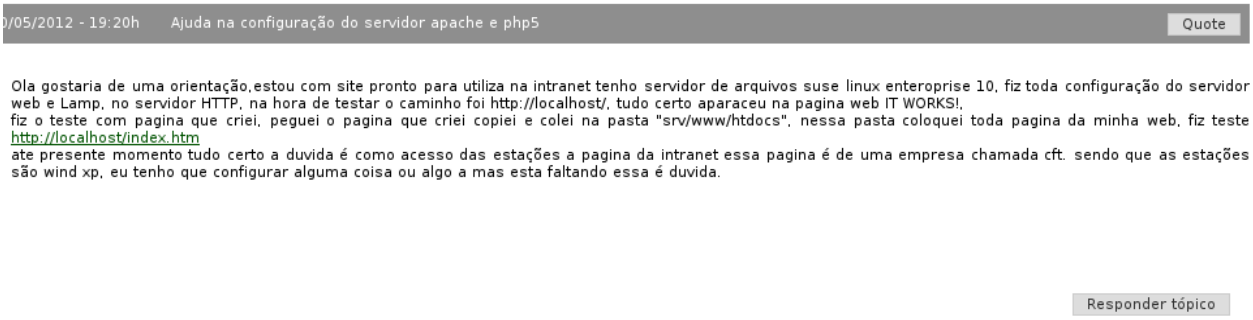


Figura 3.2: Pedido de ajuda com informações muito detalhadas

Outra forma de obter mais informações é através de anúncios de emprego. Neles é possível encontrar quais os domínios que determinado profissional precisa para atuar em um departamento da empresa.

Diversos anúncios também incluem quais softwares o possível candidato deve ter conhecimento pois fazem parte do uso cotidiano da empresa. Veja o anúncio na imagem 3.3 onde é possível obter informações sobre o sistema de banco de dados usado na empresa em questão.

3.4 Redes Sociais

As redes sociais são um dos destinos mais acessados na Internet. Não somente para permitir o intercâmbio e troca de experiência entre pessoas, muitas redes sociais se tornaram

Gerente de T.I. (v603232)

Código da vaga:	v603232
Nível hierárquico:	Pleno
Local:	Cajamar / SP / BR
Data de expiração:	19 de Setembro de 2012

Principais Responsabilidades
- Desenvolver e implementar estratégias para o ambiente de tecnologia da informação e de telecom visando atender as necessidades da América do Sul.
- Contribuir para o alinhamento da área de tecnologia da informação (TI) com as estratégias de negócio da empresa, prestando suporte nos processos de mudanças.
- Manter a rede corporativa de dados compatível com as demandas atuais das áreas de negócio continentais e globais da empresa, efetuar constantes revisões tecnologias visando aprimoramento contínuo dos serviços de telecomunicação.
- Gerenciar técnica e administrativamente os projetos dos sistemas de informação, de infra estrutura e de telecomunicação, garantindo que os dados possam ser acessados pelos clientes internos para atividades operacionais e tomada de decisão.
- Manter contato com clientes internos e externos, participando do planejamento de mudanças nos negócios da empresa e fornecendo o suporte necessário para a melhoria na utilização dos recursos.

Requisitos
- Ensino Superior Concluído;
- Mandatório Inglês avançado/ fluente (espanhol será considerado um diferencial);
- Gerenciamento de Equipes;
- Conhecimentos gerais em:
1. ERP, preferencialmente Oracle/JDE Enterprise One Versão 8. Ou acima (ideal versão 9)
2. Plataforma Windows : Banco de Dados (SQL-2005 ;SQL-2008);Exchange 2003 ou acima (desejável versão 2010); Active Directory;
3. Aplicaçoes Web e a respectiva infra-estrutura de segurança ;
4. Redes de Dados (locais e remotas) , de Voz e de Imagem

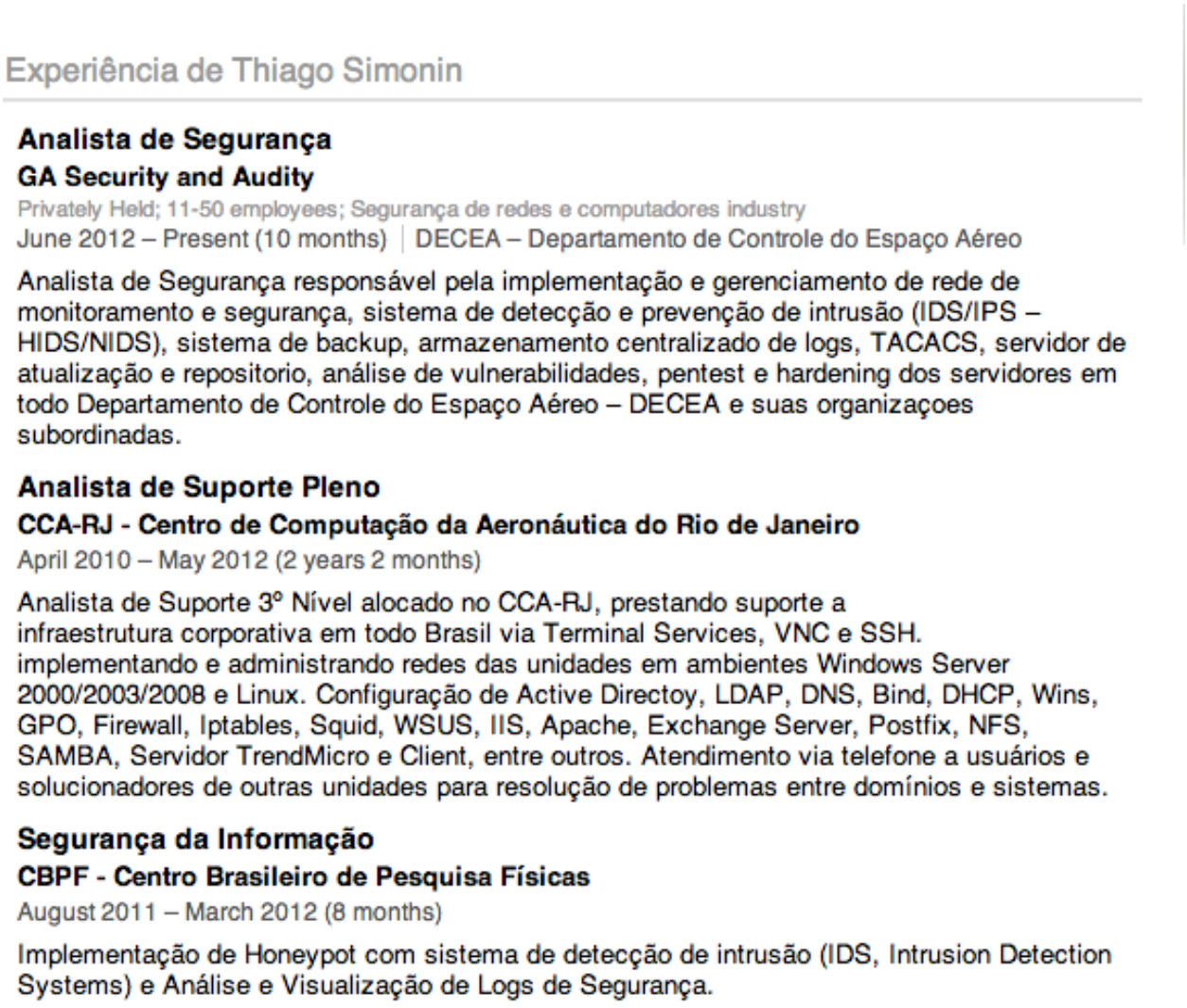
Figura 3.3: Anúncio de emprego com informações de diversos sistemas internos

mídias sociais de maneira a permitir um livre intercâmbio de informações.

Algumas redes ainda, possuem recursos como a criação de grupo ou comunidades de pessoas com interesses em comum e até de pessoas que trabalham no mesmo lugar.

Dessa maneira essas redes ou mídias sociais fornecem um imenso portfólio de informações sobre cargos, pesquisas, formação acadêmica entre outras informações. Veja na imagem 3.4

um exemplo.



Experiência de Thiago Simonin

Analista de Segurança
GA Security and Audity
Privately Held; 11-50 employees; Segurança de redes e computadores industry
June 2012 – Present (10 months) | DECEA – Departamento de Controle do Espaço Aéreo

Analista de Segurança responsável pela implementação e gerenciamento de rede de monitoramento e segurança, sistema de detecção e prevenção de intrusão (IDS/IPS – HIDS/NIDS), sistema de backup, armazenamento centralizado de logs, TACACS, servidor de atualização e repositorio, análise de vulnerabilidades, pentest e hardening dos servidores em todo Departamento de Controle do Espaço Aéreo – DECEA e suas organizaçoes subordinadas.

Analista de Suporte Pleno
CCA-RJ - Centro de Computação da Aeronáutica do Rio de Janeiro
April 2010 – May 2012 (2 years 2 months)

Analista de Suporte 3º Nível alocado no CCA-RJ, prestando suporte a infraestrutura corporativa em todo Brasil via Terminal Services, VNC e SSH. implementando e administrando redes das unidades em ambientes Windows Server 2000/2003/2008 e Linux. Configuração de Active Directoy, LDAP, DNS, Bind, DHCP, Wins, GPO, Firewall, Iptables, Squid, WSUS, IIS, Apache, Exchange Server, Postfix, NFS, SAMBA, Servidor TrendMicro e Client, entre outros. Atendimento via telefone a usuários e solucionadores de outras unidades para resolução de problemas entre domínios e sistemas.

Segurança da Informação
CBPF - Centro Brasileiro de Pesquisa Físicas
August 2011 – March 2012 (8 months)

Implementação de Honeypot com sistema de detecção de intrusão (IDS, Intrusion Detection Systems) e Análise e Visualização de Logs de Segurança.

Figura 3.4: Perfil do LinkedIn de um ex-aluno de Iniciacao Cientifica

Outro exemplo é através de pesquisas e trabalhos apresentados em congressos. Estes podem conter informações privilegiadas. Trabalhos geralmente expostos em sites como o *slideshare*, paginas pessoais ou prototipo de sistemas podem apresentar informações muito relevantes. Veja na imagem 3.5 informações que ficaram expostas por diversos meses na internet antes de serem retiradas.

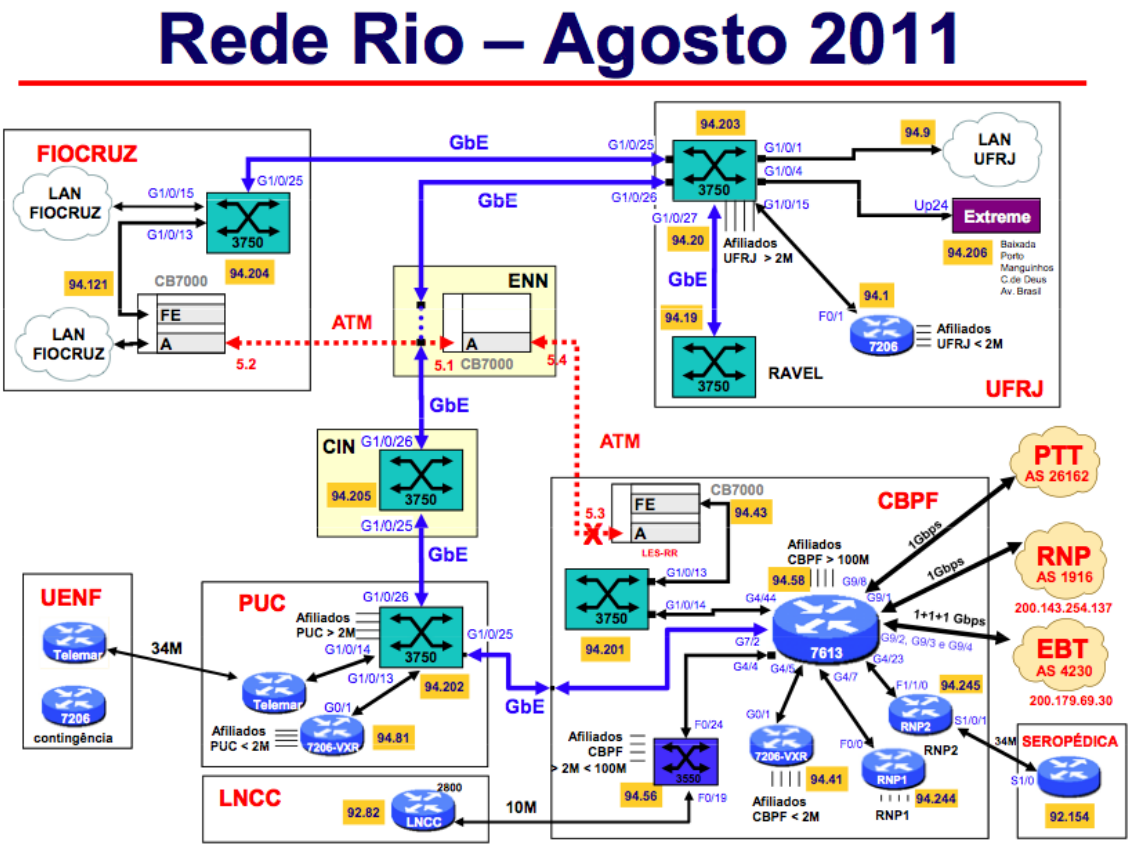


Figura 3.5: Diagrama disponivel em prototipo de Repositório Looking Glass

Capítulo 4

Mapeamento

O mapeamento de uma aplicação web é um processo divido em fases bem definidas. Esse processo é realizado de acordo com o diagrama 4.1.

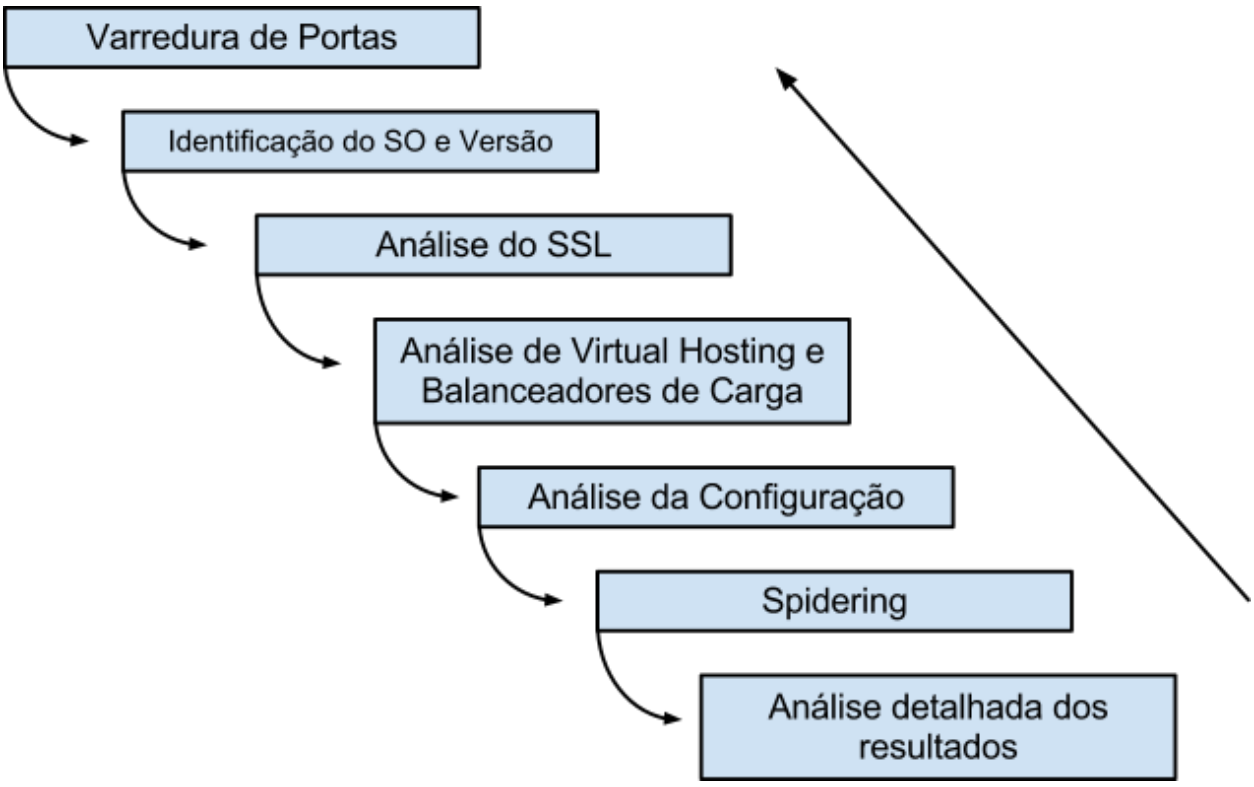


Figura 4.1: Diagrama com fases do processo de mapeamento

Assim podemos notar que as fases sempre são retro-alimentadas pelas novas informações obtidas e o processo é cíclico.

4.1 Mapeamento de portas

Varreduras de portas, tradicionalmente, têm por objetivo identificar o estado das portas dos sistemas alvo, isto é, determinar se as portas dos sistemas alvos estão abertas, fechadas e ou filtradas. A varredura de portas identifica quais portas aceitam e interpretam pacotes e, por isto, podem ser pontos de entrada para ataques. Este tipo de varredura frequentemente está associada à identificação de serviços providos, embora não precise estar necessariamente.

Com a identificação dos serviços providos pelos sistemas alvo, é possível realizar varreduras exploratórias específicas para cada serviço, a fim de obter mais informações sobre o sistema, o serviço ou até os seus usuários.

Outra atividade importante desta etapa é a identificação de sistemas operacionais (*OS fingerprinting*), que, em primeira análise, não teria relação com a varredura de portas. No entanto, como a identificação de sistemas operacionais funciona de forma similar à varredura de portas, é comum executar as duas operações na mesma etapa. A identificação é feita através da análise comportamental das respostas recebidas quando pacotes com características especiais são enviados para portas abertas e fechadas, valendo se que as diferentes implementações das pilhas UDP, TCP e IP de cada sistema operacional respondem de forma diferente, permitindo a identificação do mesmo.

A técnica de varredura mais conhecida é a de varreduras de portas lógicas de um sistema.

Uma porta lógica é uma forma de um protocolo associar um serviço a um meio de acesso, dessa forma, por exemplo, um pacote que chega na porta 80 será associado ao serviço que está nessa porta (geralmente é o servidor web).

A importância da varredura de portas está na descobertas dos serviços habilitados e possivelmente vulneráveis que possam ser alvos de uma futura investida. Assim, o mapeamento possibilita adicionar informações às fases de reconhecimento do ambiente e criar um roteiro dos possíveis serviços que podem ser alvos de ataques. Para entender de forma clara como funciona uma varredura de portas, se faz necessário um maior entendimento de como os protocolos funcionam. No exemplo a seguir, ilustramos um protocolo de atendimento:

1. O cliente entra na loja e se dirige ao atendente;
2. O atendente o cumprimenta;
3. O cliente cumprimenta o atendente;
4. O cliente faz sua solicitação ao atendente;
5. O atendente responde a solicitação do cliente.

Podemos notar como funciona o protocolo de atendimento e é com base nisto que os protocolos de internet funcionam. As varreduras são baseadas principalmente em falhas ou atitudes inesperadas que não estão previstas no protocolo, como o caso de um cliente armado chegar na loja. Um exemplo é que para saber se um atendente está disponível basta se dirigir a ele, após ele cumprimentar o cliente podemos simplesmente ir embora. Na varredura de portas, não é diferente, como somente precisamos saber se as portas estão abertas ou

fechadas, filtradas ou não-filtradas, só precisamos cumprir o protocolo até determinarmos isto, depois podemos interromper a conexão.

Podemos notar como funciona o protocolo de atendimento e é com base nisto que os protocolos de internet funcionam. As varreduras são baseadas principalmente em falhas ou atitudes inesperadas que não estão previstas no protocolo, como o caso de um cliente armado chegar na loja. Um exemplo é que para saber se um atendente está disponível basta se dirigir a ele, após ele cumprimentar o cliente podemos simplesmente ir embora. Na varredura de portas, não é diferente, como somente precisamos saber se as portas estão abertas ou fechadas, filtradas ou não-filtradas, só precisamos cumprir o protocolo até determinarmos isto, depois podemos interromper a conexão.

Acompanhe a seguir um exemplo de varredura TCP SYN com o *nmap*[1].

```
$ sudo nmap -sS -sV 200.20.94.130

Starting Nmap 6.25 ( http://nmap.org ) at 2013-03-22 09:52 BRT
Nmap scan report for 200.20.94.130
Host is up (0.062s latency).
Not shown: 991 closed ports
PORT     STATE    SERVICE        VERSION
25/tcp   filtered smtp
80/tcp   open     http           Apache httpd 2.2.15 ((Fedora))
135/tcp  filtered msrpc
139/tcp  filtered netbios-ssn
443/tcp  open     ssl/http       Apache httpd 2.2.15 ((Fedora))
445/tcp  filtered microsoft-ds
593/tcp  filtered http-rpc-epmap
3306/tcp open     mysql          MySQL 5.1.47
4444/tcp filtered krb524

Service detection performed. Please report any incorrect results at http://nmap.org/subm
Nmap done: 1 IP address (1 host up) scanned in 125.11 seconds
```

4.2 Identificação do SO e Versão

A identificação do Sistema Operacional é importante para determinar possíveis vulnerabilidades no sistema anfitrião. Embora não seja esse o foco, essas vulnerabilidades podem facilitar a descoberta de vulnerabilidades na aplicação web.

Acompanhe a seguir um exemplo com o *nmap*.

```
$ sudo nmap -sS -O 200.20.94.130
```

[1]O *Network Mapper* é um softwares mais usados para varreduras de portas. Mais informações sobre varreduras podem ser obtidas em: `http://nmap.org/book/man-port-scanning-techniques.html`

```
Starting Nmap 6.25 ( http://nmap.org ) at 2013-03-22 09:56 BRT
Nmap scan report for 200.20.94.130
Host is up (0.074s latency).
Not shown: 952 closed ports, 45 filtered ports
PORT     STATE SERVICE
80/tcp   open  http
443/tcp  open  https
3306/tcp open  mysql
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.24 - 2.6.36
Network Distance: 12 hops

OS detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 60.40 seconds
```

4.3 NSE Scripts

O *nmap* também possui a possibilidade de uso de scripts, chamado *Nmap Scripting Engine*. Esse modo extende o uso da ferramente para algo além de mapear redes e enumeração de serviços, ele aplica funções de bruteforce, ataques DoS e até mesmo descoberta de vulnerabilidades. Os scripts podem ser escritos e compartilhados por qualquer usuário no objetivo de automatizar as tarefas diárias.

A linguagem utilizada é a Lua, pequena e flexível, geralmente utilizada para expandir funções de programas e jogos. Esse post não mostrará como desenvolver os scripts, mas só como eles são utilizados.

Os scripts por padrão ficam no diretório `/usr/local/share/nmap/scripts` com a extensão `.nse`.

4.3.1 Categorias de scripts

Auth: Scripts voltados para tentativa de autenticação ou bypass da mesma.

Broadcast: Realiza uma varredura de broadcast na rede atrás de hosts não listados por varreduras normais.

Brute: Usam técnicas de força-bruta para realizar autenticação de serviços.

Default: São os scripts padrões do Nmap, utilizado com a opção -sC ou –scripts=default. As opções serão retratadas a seguir.

Discovery: Esses scripts realizam varreduras de descoberta na rede, como cabeçalhos HTML, detalhes do serviço SAMBA, etc.

DoS: Scripts utilizados para realizar negação de serviço no alvo.

External: Utilizam aplicações de terceiros, como consultas ao banco de dados do whois.

Fuzzer: Envia pacotes com dados randômicos para o alvo, a procura de bugs e vulnerabilidades.

Intrusive: Não são considerados seguros pois podem derrubar o alvo ou invádi-lo.

Malware: Procura no alvo malwares ou backdoors.

Safe: São considerados scrips que não derrubam serviços, não utilizam toda a banda e não gera nenhum risco ao alvo.

Version: Scripts para detectar versões de serviços específicos, como o Skype. Só podem ser usados cada o tipo de scan -sV for requerido.

Vuln: Usados para procurar vulnerabilidades. Só são gerados relatórios se elas são encontradas.

4.3.2 Fases

Prerule scripts: São executados antes do Nmap recolher qualquer informação sobre o alvo.

Host scripts: Ocorre durante o processo de varredura, logo depois da varredura de portas, serviços e sistema operacional.

Service scripts: São executados em serviços específicos no host.

Postrule scripts: Esses scripts rodam depois que toda a varredura for feita. Geralmente auxiliam na formatação de relatórios.

Sabendo das informações acima, vamos fazer uma verificação dos métodos do protocolo HTTP que são permitidos no www.cbpf.br.

```
$ nmap --script=http-methods.nse --script-args http-methods.retest=1 -p80 www.cbpf.br

Starting Nmap 6.25 ( http://nmap.org ) at 2013-03-22 10:10 BRT
Nmap scan report for www.cbpf.br (152.84.253.9)
Host is up (0.71s latency).
rDNS record for 152.84.253.9: cbpfsu6.cat.cbpf.br
PORT   STATE SERVICE
80/tcp open  http
| http-methods: GET HEAD POST OPTIONS
```

```
| GET / -> HTTP/1.1 200 OK
|
| HEAD / -> HTTP/1.1 200 OK
|
| POST / -> HTTP/1.1 200 OK
|
|_OPTIONS / -> HTTP/1.1 200 OK

Nmap done: 1 IP address (1 host up) scanned in 8.01 seconds
```


Capítulo 5

Identificação da Aplicação

5.1 Infraestrutura Web

A identificação da infraestrutura web é importante para poder determinar por onde os pacotes com as requisições podem ter passado com a finalidade de determinar alguma forma de tratamento intermediaria.

Também é importante pois pode ser possível encontrar vulnerabilidades nos equipamentos ou softwares instalados neles.

5.1.1 Load Balancers

O uso de Balanceadores de Carga ou *Load Balancers* é uma realidade muito comum hoje em aplicações que necessitam de alta disponibilidade. Seu uso é uma questão que deve ser investigada pois pode causar diversas inconsistências nos resultados de diversos testes, principalmente quando os servidores que servem a aplicação possuem características, como: gerência de patches, correção de bugs, versões de software e arquitetura diferentes.

Existem diversas implementações para prover características de balanceamento de carga. Existem balanceamentos de carga com uso de DNS ou com o uso de equipamentos próprios para esse fim.

Quando o balanceamento é realizado através do sistema de DNS, é relativamente simples determinar os sistemas que fazem parte do grupo de balanceamento.

Veja a consulta, a seguir, realizada com o *dig*:

```
$ dig mail.google.com

; <<>> DiG 9.8.3-P1 <<>> mail.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18539
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 0
```

```
;; QUESTION SECTION:
;mail.google.com. IN A

;; ANSWER SECTION:
mail.google.com. 296457 IN CNAME googlemail.l.google.com.
googlemail.l.google.com. 85 IN A 74.125.234.54
googlemail.l.google.com. 85 IN A 74.125.234.53

;; Query time: 630 msec
;; SERVER: 187.100.246.251#53(187.100.246.251)
;; WHEN: Fri Mar 22 10:20:54 2013
;; MSG SIZE  rcvd: 92
```

Além do balanceamento através dos sistemas de DNS, é possível realizar o balanceamento através de equipamentos específicos. Esses equipamentos possuem métodos de balanceamento um pouco mais eficientes, como, por exemplo:

Least Connections -> encaminha a conexão para o servidor com menos conexões

Round Robin -> encaminha a conexão de forma dinâmica

Weighted Round Robin -> encaminha a conexão de forma dinâmica, atribuindo pesos para os servidores no grupo

O interessante dessas formas é que o pacote chegue ao sistema de destino, sem a intervenção do balanceador. Dessa forma com o uso da ferramenta hping3, podemos determinar através do campo IPID. Acompanhe o exemplo a seguir, para mais detalhes.

```
$ sudo hping3 www.microsoft.com -S -p 80
HPING www.microsoft.com (en1 65.55.57.27): S set, 40 headers + 0 data bytes
→ len=44 ip=65.55.57.27 ttl=238 DF id=26820 sport=80 flags=SA seq=0 win=8190 rtt=214.7 m
len=44 ip=65.55.57.27 ttl=237 DF id=44411 sport=80 flags=SA seq=1 ...
→ len=44 ip=65.55.57.27 ttl=237 DF id=31028 sport=80 flags=SA seq=2 ...
len=44 ip=65.55.57.27 ttl=237 DF id=18317 sport=80 flags=SA seq=3 ...
→ len=44 ip=65.55.57.27 ttl=238 DF id=34635 sport=80 flags=SA seq=4 ...
len=44 ip=65.55.57.27 ttl=238 DF id=61539 sport=80 flags=SA seq=5 ...
→ len=44 ip=65.55.57.27 ttl=237 DF id=38350 sport=80 flags=SA seq=6 ...
len=44 ip=65.55.57.27 ttl=238 DF id=4466 sport=80 flags=SA seq=7 ...
len=44 ip=65.55.57.27 ttl=237 DF id=35175 sport=80 flags=SA seq=8 ...
→ len=44 ip=65.55.57.27 ttl=237 DF id=48555 sport=80 flags=SA seq=9 ...
→ len=44 ip=65.55.57.27 ttl=237 DF id=50443 sport=80 flags=SA seq=10 ...
len=44 ip=65.55.57.27 ttl=237 DF id=28393 sport=80 flags=SA seq=11 ...
len=44 ip=65.55.57.27 ttl=237 DF id=55471 sport=80 flags=SA seq=12 ...
→ len=44 ip=65.55.57.27 ttl=237 DF id=52008 sport=80 flags=SA seq=13 ...
```



```
len=44 ip=65.55.57.27 ttl=237 DF id=37115 sport=80 flags=SA seq=14 ...
len=44 ip=65.55.57.27 ttl=238 DF id=41838 sport=80 flags=SA seq=15 ...
len=44 ip=65.55.57.27 ttl=238 DF id=20646 sport=80 flags=SA seq=16 ...
len=44 ip=65.55.57.27 ttl=237 DF id=33975 sport=80 flags=SA seq=17 ...
len=44 ip=65.55.57.27 ttl=238 DF id=35440 sport=80 flags=SA seq=18 ...
len=44 ip=65.55.57.27 ttl=237 DF id=34927 sport=80 flags=SA seq=19 ...
len=44 ip=65.55.57.27 ttl=238 DF id=18502 sport=80 flags=SA seq=20 ...
len=44 ip=65.55.57.27 ttl=238 DF id=27422 sport=80 flags=SA seq=21 ...
len=44 ip=65.55.57.27 ttl=237 DF id=10820 sport=80 flags=SA seq=22  ...
^C
--- www.microsoft.com hping statistic ---
23 packets tramitted, 23 packets received, 0% packet loss
round-trip min/avg/max = 196.1/236.0/361.5 ms
```

Porém, nem sempre o balanceador permite tal interação. Muitas vezes é o próprio balanceador que estabelece a conexão com o cliente, assim a saída do sistema seria:

```
$ sudo hping3 mail.google.com -S -p 80
HPING mail.google.com (ppp0 74.125.234.118): S set, 40 headers + 0 data bytes
len=44 ip=74.125.234.118 ttl=52 id=18861 sport=80 flags=SA seq=0 ...
len=44 ip=74.125.234.118 ttl=52 id=18862 sport=80 flags=SA seq=1 ...
len=44 ip=74.125.234.118 ttl=52 id=18863 sport=80 flags=SA seq=2 ...
len=44 ip=74.125.234.118 ttl=52 id=18864 sport=80 flags=SA seq=2 ...
len=44 ip=74.125.234.118 ttl=52 id=18865 sport=80 flags=SA seq=3 ...
len=44 ip=74.125.234.118 ttl=52 id=18866 sport=80 flags=SA seq=4 ...
len=44 ip=74.125.234.118 ttl=52 id=18868 sport=80 flags=SA seq=5 ...
len=44 ip=74.125.234.118 ttl=52 id=18869 sport=80 flags=SA seq=6 ...
^C
--- mail.google.com hping statistic ---
8 packets tramitted, 8 packets received, 0% packet loss
round-trip min/avg/max = 220.2/307.9/649.7 ms
```

Além de balancear a carga entre os servidores, por vezes é necessário que o balanceador seja responsável pelos métodos de persistência de conexão. Este ocorre pois diversas aplicações necessitam, por exemplo, de autenticação ou de controle de sessão.

Os métodos de persistência mais utilizados são:

- Persistência por endereço IP de origem
- Persistência baseada em Cookie
- Persistência baseada no hash da URI

5.1.2 WAF

Um firewall de aplicação web (do inglês, *Web Application Firewall* - WAF) é um aparelho, plugin de servidor, ou filtro que aplica um conjunto de regras para uma conexão HTTP. Geralmente, essas regras abrangem ataques comuns como Cross-site Scripting (XSS) e SQL Injection. Ao personalizar as regras para a aplicação em questão, muitos ataques podem ser identificadas e bloqueadas. O esforço para realizar esta personalização pode ser significativo e precisa ser mantido conforme a aplicação é modificada.

A detecção de ferramentas WAF também pode ser realizada com o uso do nmap. Acompanhe alguns exemplos a seguir.

NSE HTTP-WAF-DETECT

```
$ nmap -p80,443 --script http-waf-detect www.phpids.org

Starting Nmap 6.01 ( http://nmap.org ) at 2012-10-24 18:16 BRST
Nmap scan report for www.phpids.org (188.40.98.186)
Host is up (0.23s latency).
rDNS record for 188.40.98.186: nepal1.itratos.net
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https
| http-waf-detect: IDS/IPS/WAF detected:
|_www.phpids.org:443/?p4yl04d3=<script>alert(document.cookie)</script>

Nmap done: 1 IP address (1 host up) scanned in 16.22 seconds
```

NSE HTTP-WAF-FINGERPRINT

```
$ nmap --script=http-waf-fingerprint <targets>

PORT   STATE SERVICE REASON
80/tcp open  http    syn-ack
| http-waf-fingerprint:
|   Detected WAF
|_    BinarySec version 3.2.2
```

LBMAP

O lbmap[1] é uma prova de conceito (PoC) e também pode ser utilizada para esse fim. Acompanhe um exemplo a seguir:

[1] https://github.com/wireghoul/lbmap

```
mac159275:wireghoul-lbmap-ac964d1 manoel.junior$ ./lbmap login.globo.com
lbmap - http fingerprinting tool
Eldar "Wireghoul" Marcussen - Scanning login.globo.com
$VAR1 = 'signature';
$VAR2 = '01A2A2--99--99BCA2--A2A2A2A2L3BCA2A2A2BCA2BCBCA2BCBCA2A099--';
$VAR3 = 'webserver';
$VAR4 = {
          'Apache' => 22
        };
```

Diversos outros exemplos podem ser encontrados atraves, principalmente de apresentacoes em eventos e congressos especializados[2].

5.1.3 Servidores Web

APROF

A ferramenta *aprof* faz parte da suite da ferramenta *lbmap*, abordada anteriormente. Ela permite, além de identificar o servidor web, também permite identificar os módulos habilitados na instância testada.

Veja um exemplo a seguir:

```
$ ./aprof www.cbpf.br
aprof - Apache profiler script v 0.2
Written by Eldar "Wireghoul" Marcussen - http://www.justanotherhacker.com
Analysing: www.cbpf.br/
Server signature: Apache
Attempting to identify modules:
mod_cgi detected
mod_info (restricted access) detected
mod_php detected
Checking what extensions have php bindings:
.php ... in use
.php3 ... in use
.php4 ... in use
.php5 ... not in use
.phtml ... not in use
.phps ... not in use
mod_status (restricted access) detected
Done!
```

[2] www.defcon.org/images/defcon-16/dc16-presentations/defcon-16-henrique.pdf - http://media.blackhat.com/bh-us-12/Briefings/Ristic/BH_US_12_Ristic_Protocol_Level_WP.pdf

NMAP

Como já sabemos, a ferramenta NMAP também é capaz de identificar com precisão diversas versões de diversos servidores web. Acompanhe, a seguir, um exemplo.

```
$ sudo nmap -sV -p80,443 www.cbpf.br

Starting Nmap 6.25 ( http://nmap.org ) at 2013-03-22 11:19 BRT
Nmap scan report for www.cbpf.br (152.84.253.9)
Host is up (0.071s latency).
rDNS record for 152.84.253.9: cbpfsu6.cat.cbpf.br
PORT    STATE SERVICE VERSION
80/tcp  open  http    Apache
443/tcp open  http    Apache
Service Info: Host: www.cbpf.br

Service detection performed. Please report any incorrect results at http://nmap.org/subm
Nmap done: 1 IP address (1 host up) scanned in 54.65 seconds
```

5.1.4 Servidores de Banco de Dados

NMAP

Através da mesma maneira que identificamos servidores web, podemos identificar servidores de banco de dados. Veja o exemplo a seguir:

```
$ sudo nmap -sS -sV 200.20.94.130

Starting Nmap 6.25 ( http://nmap.org ) at 2013-03-22 09:52 BRT
Nmap scan report for 200.20.94.130
Host is up (0.062s latency).
Not shown: 991 closed ports
PORT     STATE    SERVICE        VERSION
25/tcp   filtered smtp
80/tcp   open     http           Apache httpd 2.2.15 ((Fedora))
135/tcp  filtered msrpc
139/tcp  filtered netbios-ssn
443/tcp  open     ssl/http       Apache httpd 2.2.15 ((Fedora))
445/tcp  filtered microsoft-ds
593/tcp  filtered http-rpc-epmap
3306/tcp open     mysql          MySQL 5.1.47
4444/tcp filtered krb524

Service detection performed. Please report any incorrect results at http://nmap.org/subm
Nmap done: 1 IP address (1 host up) scanned in 125.11 seconds
```


5.2 Versões e Configurações

5.2.1 Google Hacking

O GHDB ou *Google Hacking Database* é um projeto que visa colecionar padrões de busca que, ao serem usadas no buscador do Google podem revelar informações importantes sobre determinado domínio.

Ele pode ser acessado através da URL `http://www.hackersforcharity.org/ghdb/`. No site, é possível encontrar de forma organizada e categorizada como pesquisar no buscador do Google determinadas informações.

Um exemplo de informações interessantes que podem ser obtidas com buscas usando padrões no GHDB é como encontrar arquivos com possíveis nomes de usuários. Basta ir em "Files containing usernames", e procurar o arquivo de interesse. Na terceira coluna, é possível obter a sintaxe de procura no buscador do Google.

Embora o site seja bem organizado, existe uma iniciativa bem parecida que se chama "Google Dorks". Ela pode ser acessada através da URL `http://www.exploit-db.com/google-dorks/`. No momento da redação desse resumo, ela estava mais atualizada que a anterior.

5.2.2 Análise do SSL

A análise dos certificados SSL pode ser útil para determinar possíveis vulnerabilidades tanto na aplicação, quanto no certificado emitido.

Veja, na imagem 5.1, um relatório[3] da ferramenta SSL Labs[4], da Qualys.

Embora não seja possível ver os detalhes do relatório na imagem, já podemos examinar que o certificado SSL está sucetível ao Ataque BEAST[5] e que ele não atente completamente nenhum dos quatro requisitos avaliados.

5.3 Verificando Virtual Hosts

Um *Virtual Host* é uma maneira de servir múltiplos sites em um mesmo servidor físico. Essa técnica é utilizada na configuração do servidor web, não sendo portanto, uma virtualização. Esta técnica, geralmente, é utilizada para alocar melhor os recursos e diminuir custos com a hospedagem de sites, mas também apresenta problemas quanto a segurança, pois em caso de uma vulnerabilidade no servidor, todos os sites estarão espostos a ela.

Existem duas formas de realizar *Virtual Hosting*. Elas são através de um único endereço IP que serve todos os sites ou vários endereços IP, um endereço para cada site, todos associados no mesmo servidor.

[3] Disponívelem:`https://www.ssllabs.com/ssltest/analyze.html?d=p0m1.cat.cbpf.br`

[4] `www.ssllabs.com`

[5] Mais informações em: `https://community.qualys.com/blogs/securitylabs/2011/10/17/mitigating-the-beast-attack-on-tls`

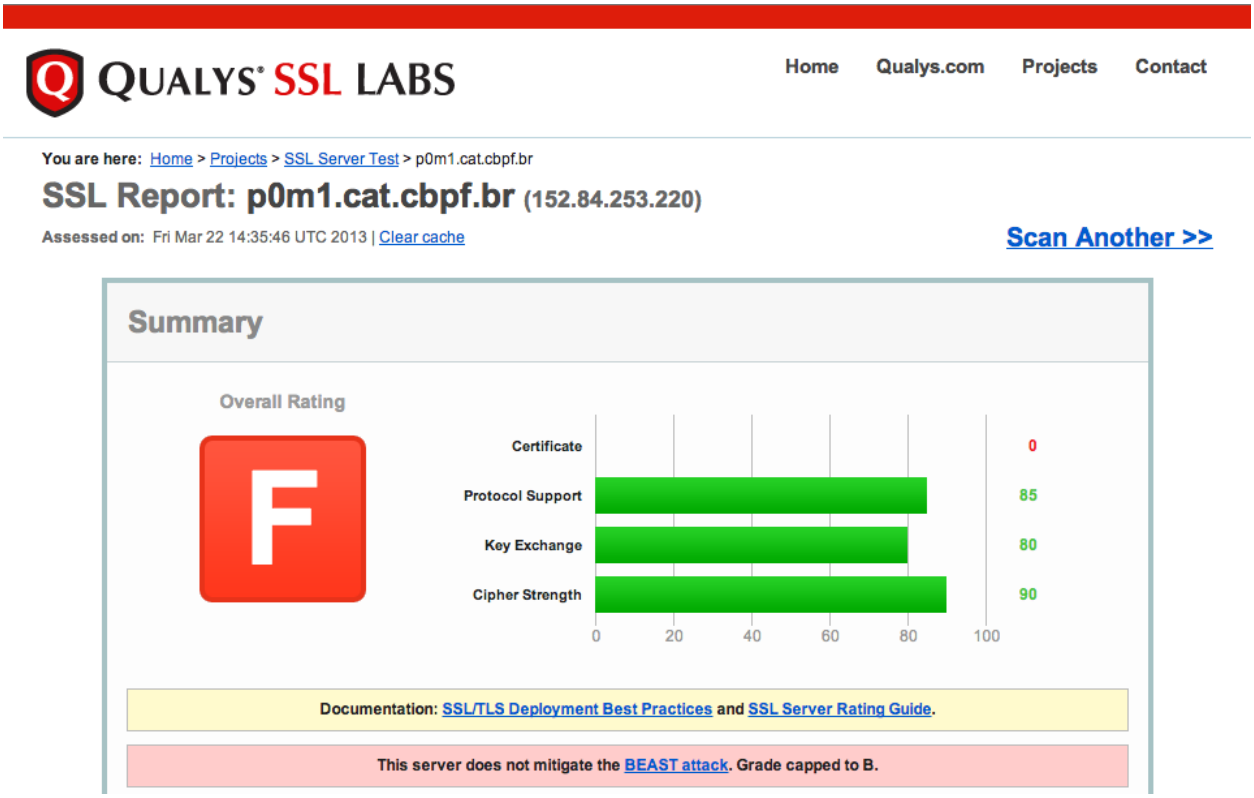


Figura 5.1: Análise do Certificado SSL do email do CBPF

Para realizar a identificação, basta fazer consultas de DNS. Podemos realizar a ferramenta online *Robtex*[6] para agilizar a consulta. Acompanhe o exemplo na imagem 5.2.

5.4 Usando Nikto

Nikto[7] é um scanner de servidor web que realiza testes abrangentes contra servidores para identificar vários itens, incluindo mais de 6.500 arquivos/CGIs potencialmente perigosos, cheques de versões desatualizadas de mais de 1250 servidores, e os problemas específicos de versão em mais de 270 servidores. Ele também verifica a existência de itens de configuração do servidor, tais como a presença de vários arquivos de índice, opções do servidor HTTP, e vai tentar identificar o servidores web e o software instalado. Itens de verificação e plugins são atualizados com freqüência e podem ser atualizado automaticamente.

Nikto não foi concebido como uma ferramenta muito furtiva. Ele irá testar um servidor web no tempo mais rápido possível, e é bastante óbvio em arquivos de log. No entanto, há suporte para *Libwhisker*[8], que usa métodos anti-IDS no caso de você querer experimentá-lo

[6]Disponível em: `http://robtex.com/`
[7]Disponível em: `http://www.cirt.net/nikto2`
[8]Disponível em: `http://sourceforge.net/projects/whisker/`

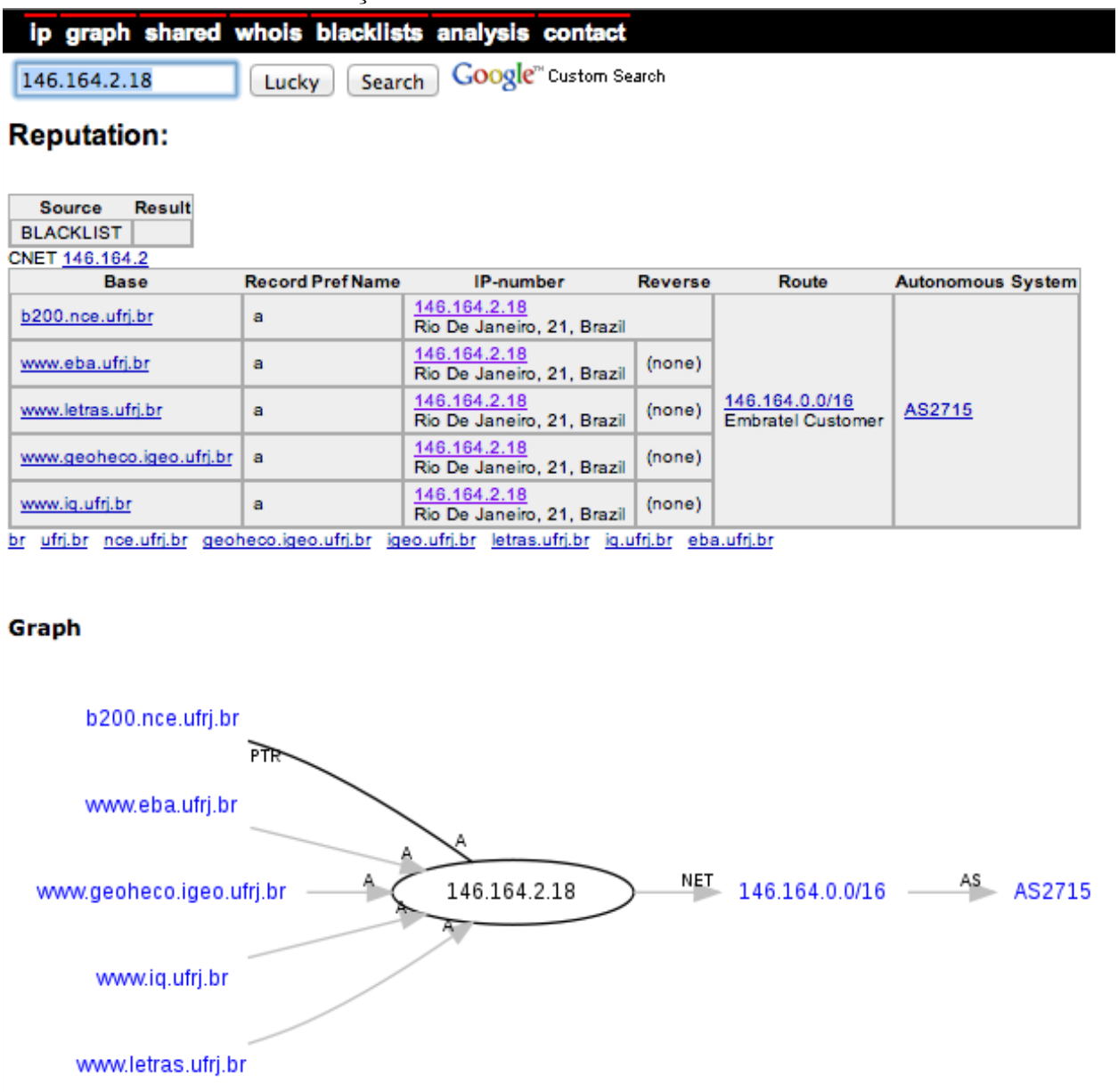


Figura 5.2: Endereço IP servindo varios domínios

(ou testar seu sistema IDS).

Nem toda verificação é um problema de segurança, embora a maioria seja. Existem alguns itens que são apenas "informativos"verificações desse tipo servem para averiguar coisas que podem não ter uma falha de segurança, mas o webmaster ou o engenheiro de segurança pode não saber estão presentes no servidor. Esses itens são geralmente marcados apropriadamente na informação impressa.

Acompanhe a seguir um exemplo de uso da ferramenta *Nikto* em sua configuração padrão.

```
$ sudo ./nikto.pl -host www.cbpf.br
- Nikto v2.1.5
---------------------------------------------------------------------------
+ Target IP:          152.84.253.9
+ Target Hostname:    www.cbpf.br
+ Target Port:        80
+ Start Time:         2013-03-22 12:06:16 (GMT-3)
---------------------------------------------------------------------------
+ Server: Apache
+ The anti-clickjacking X-Frame-Options header is not present.
+ OSVDB-637: Enumeration of users is possible by requesting ~username (responds with 'Fo
+ Server leaks inodes via ETags, header found with file /favicon.ico, inode: 75112357, s
+ Allowed HTTP Methods: GET, HEAD, POST, OPTIONS
+ /global.asa: The global.asa file was retrieved, which may contain sensitive informatio
+ OSVDB-3092: /_vti_pvt/deptodoc.btr: FrontPage file found. This may contain useful info
+ OSVDB-473: /_vti_pvt/botinfs.cnf: FrontPage file found. This may contain useful inform
+ OSVDB-473: /_vti_pvt/bots.cnf: FrontPage file found. This may contain useful informati
+ OSVDB-473: /_vti_pvt/service.cnf: Contains meta-information about the web server Remov
+ OSVDB-473: /_vti_pvt/writeto.cnf: Contains information about form handler result files
+ OSVDB-3268: /pdf/: Directory indexing found.
+ OSVDB-3268: /dir/: Directory indexing found.
+ OSVDB-3092: /dir/: This might be interesting...
+ Cookie PHPSESSID created without the httponly flag
+ Retrieved x-powered-by header: PHP/5.3.8
+ 6544 items checked: 4961 error(s) and 15 item(s) reported on remote host
+ End Time:           2013-03-22 12:24:34 (GMT-3) (1098 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

5.5 Spidering

O nosso objetivo é o de criar um mapa da aplicação com todos os pontos de acesso para a aplicação. Isto será útil para a segunda fase do teste de penetração. Você pode usar uma ferramenta como o *wget*[9] para recuperar todas as informações publicadas pela aplicação.

Acompanhe o exemplo a seguir, onde mapeamos o cabeçalho:

```
$ wget -S www.cbpf.br
--2013-03-22 12:08:51--  http://www.cbpf.br/
Resolving www.cbpf.br (www.cbpf.br)... 152.84.253.9
Connecting to www.cbpf.br (www.cbpf.br)|152.84.253.9|:80... connected.
HTTP request sent, awaiting response...
```

[9]Disponível em: `http://www.gnu.org/software/wget/`


```
  HTTP/1.1 200 OK
  Date: Fri, 22 Mar 2013 15:07:58 GMT
  Server: Apache
  Accept-Ranges: bytes
  Content-Length: 266
  Keep-Alive: timeout=5, max=100
  Connection: Keep-Alive
  Content-Type: text/html
Length: 266 [text/html]
Saving to: ‘index.html’

100%[======================================>] 266         --.-K/s   in 0s

2013-03-22 12:08:52 (10.6 MB/s) - ‘index.html’ saved [266/266]
```

O ideal ao utilizar o *wget* e especificar o parâmetro "-r" de modo a realizar consultas recursivas.

Mais adiante no capítulo 7, veremos como executar o *Spidering* de forma mais automatizada.

5.6 Fluxograma de funcionamento

Nessa etapa, devemos focar na identificação dos componentes da aplicação. Mapear o fluxo da aplicação é importante para prover um melhor entendimento de como a aplicação funciona inteiramente. O fluxo da aplicação deverá deixar de forma clara a relação entre as partes da aplicação, como eles interagem e identificar possíveis interfaces de ataque.

Veja um exemplo de mapeamento de fluxo na imagem 5.3.

5.7 Identificação dos pontos de entrada de informação

A enumeração de um aplicativo e sua superfície de ataque é um passo chave antes que qualquer teste completo possa ser realizado, pois permite identificar as áreas susceptíveis a falhas. Esta seção tem como objetivo ajudar a identificar e mapear essas áreas dentro da aplicação que deve ser investigada, uma vez que a enumeração e o mapeamento foram concluídos.

Antes do início de qualquer teste, é sempre importante obter um bom entendimento da aplicação e como o usuário ou browser se comunica com ele. Ao navegar pela aplicação, preste atenção especial a todas as solicitações HTTP (métodos GET e POST), bem como todos os parâmetros e campos de formulários que são passados para a aplicação. Além disso, preste atenção quando requisições GET são usados e quando as solicitações POST são usadas para passar parâmetros para a aplicação.

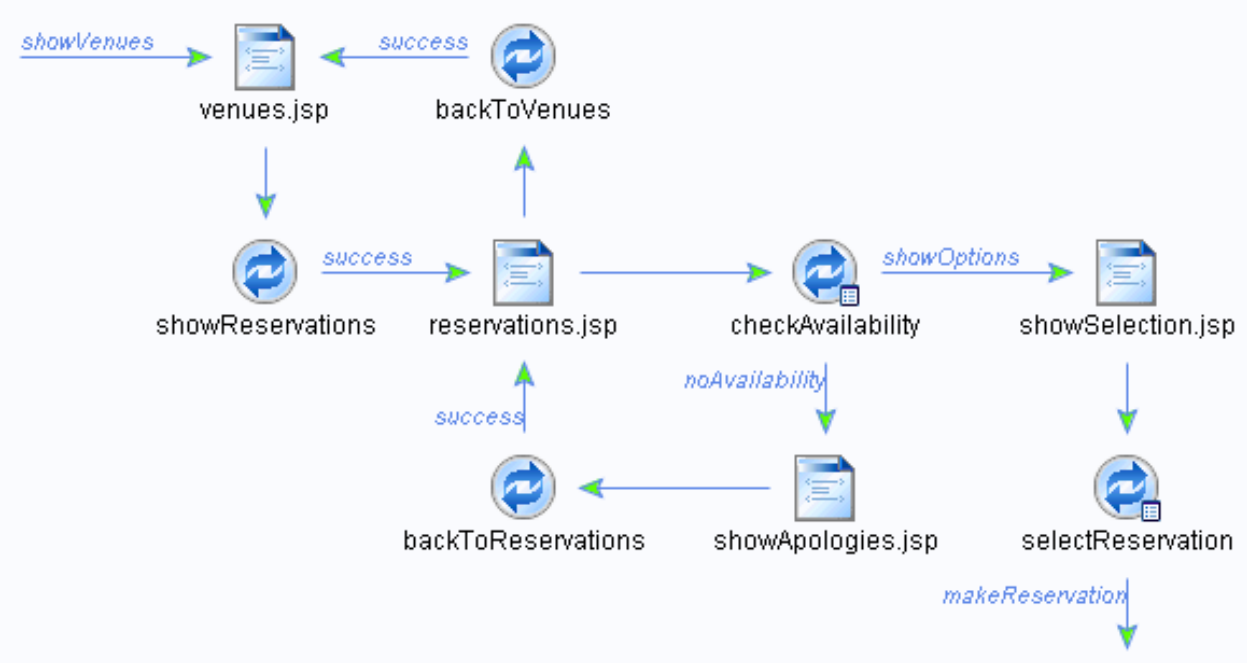


Figura 5.3: Fluxo de uma aplicação web

É muito comum ter grande parte das solicitações GET, mas quando uma informação sensível é passada, muitas vezes é feito dentro do corpo de uma solicitação POST. Note-se que para ver os parâmetros enviados em uma solicitação POST, você precisará usar uma ferramenta como um proxy de interceptação (por exemplo, *OWASP ZAP*) ou um plug-in para o navegador. Dentro da solicitação POST, também faça uma nota especial de todos os campos de formulário 'hidden" que estão sendo passados para a aplicação, pois estes geralmente contêm informações sensíveis, tais como informações de estado, quantidade de itens, o preço de itens, campos que o desenvolvedor não quer que o usuário veja ou modifique.

Uma dica muito útil é usar um *proxy de interceptação* e uma planilha para esta fase do teste. O *proxy* irá acompanhar cada pedido e resposta entre você e a aplicação. Além disso, neste momento, é comum se prender a cada solicitação e resposta de modo que se possa ver exatamente cada cabeçalho, parâmetro, etc, que está sendo passado para a aplicação e que está sendo devolvido para o usuário. Isso pode ser bastante tedioso às vezes, especialmente em grandes sites interativos (como em uma aplicação bancária). No entanto, a experiência vai lhe ensinar o que procurar, e, portanto, esta fase pode ser reduzida significativamente.

Conforme você navega pela da aplicação, tome nota de todos os parâmetros interessantes na URL, cabeçalhos personalizados, ou corpo das requisições ou respostas, e salve-os na planilha. A planilha deve incluir a página que você solicitou (que pode ser bom também para adicionar o número de pedido do*proxy*, para futura referência), os parâmetros de interesse, o tipo de solicitação (POST ou GET), se o acesso é autenticado ou não autenticado, se SSL é usado, se é parte de um processo de várias etapas, e quaisquer outras notas relevantes.

Depois de ter todas as áreas de aplicação traçadas, então você pode ir até o aplicativo e

testar cada uma das áreas que você identificou e fazer anotações para o que funcionou e o que não funcionou. O restante deste guia vai identificar como testar cada uma dessas áreas de interesse, mas esta seção devem ser realizadas antes de qualquer teste real pode começar.

A seguir, estão alguns pontos de interesse para todos as requisições e respostas. Dentro da seção de requisições, o foco é sobre os métodos GET e POST, como estes aparecem na maioria das requisições. Note-se que outros métodos, tais como PUT e DELETE, podem ser usados. Muitas vezes, estes pedidos mais raros, se forem permitido, podem expor mais vulnerabilidades.

5.7.1 Requisições

1. Identificar onde GETs são usados e onde POSTs são usados.

2. Identificar todos os parâmetros utilizados num pedido POST (estes estão no corpo do pedido).

3. Dentro da solicitação POST, preste atenção especial a quaisquer parâmetros do tipo 'hidden". Quando um POST é enviado, todos os campos do formulário (incluindo parâmetros ocultos) serão enviado no corpo da mensagem HTTP para a aplicação. Estes geralmente não são vistos, a menos que você está usando um proxy ou visualizar o código fonte HTML. Além disso, a página seguinte pode ver os dados, e o seu acesso podem ser diferente, dependendo do valor do parâmetro escondido(s).

4. Identificar todos os parâmetros utilizados em uma solicitação GET (ou seja, URL), em uma determinada sequência

5. Identificar todos os parâmetros da cadeia de consulta. Estes geralmente são em pares, como foo = bar. Observe também que muitos parâmetros podem surgir em uma sequência, como separados por um &, ,:, ou qualquer outro carácter especial ou de codificação.

6. Uma nota especial quando se trata de identificar vários parâmetros numa cadeia ou num pedido POST, é identificar todos os parâmetros necessários para executar os ataques. Você precisa identificar todos os parâmetros (mesmo que estejam codificado ou criptografados) e identificar quais são processados pela aplicação.

7. Também preste atenção a qualquer tipo cabeçalhos adicionais ou que não sejam tipicamente vistos (como debug = False).

5.7.2 Respostas

1. Identificar onde os cookies são definidos (cabeçalho Set-Cookie), modificados, ou adicionados.

2. Identificar onde há qualquer redirecionamento (código de status HTTP 300), códigos de status (400), em especial: 403 (Forbidden) e erros internos do servidor (500) durante respostas normais (ou seja, os pedidos não modificados).

3. Observe também que cabeçalhos interessantes podem ser usados. Por exemplo, 'Server: BIG-IP"indica que o site utiliza load balance. Assim, se um site possui balanceamento de carga e um servidor está configurado incorretamente, então você pode ter que fazer várias solicitações para acessar o servidor vulnerável, dependendo do tipo de balanceamento de carga utilizado.

Capítulo 6

Identificação de Vulnerabilidades

O processo de descoberta de vulnerabilidade, se assemelha ao processo de descoberta de informação, sendo que nesse momento, a aplicação web passa a receber tráfego malicioso. Logo após a descoberta da vulnerabilidade pode-se dar inicio a fase de exploração delas.

A seguir, veremos alguns métodos de identificação de vulnerabilidade e logo após seus tipos e como identificá-las.

6.1 Métodos de Identificação

O principal método de identificação de vulnerabilidades é através do uso de ferramentas automatizadas que realizam diversos testes e apontam os problemas encontrados para uma verificação manual.

Os softwares usados para realizar buscas por vulnerabilidades em aplicações Web são diferentes dos softwares utilizados para buscas por vulnerabilidades em redes ou sistemas. Isso ocorre pois as ferramentas que buscam vulnerabilidades em redes e sistemas manipulam e enviam o tráfego de rede de maneira a determinar se o sistema está vulnerável ou não com base nas respostas fornecidas pelo sistema.

No caso de uma aplicação web, o software necessita se adequar a estrutura da aplicação, além de conseguir mapear os pontos onde seria possível inserir informações na aplicação. Além disso, com base nas informações recolhidas, ele deverá selecionar os possíveis ataques e executá-los, observando sua resposta para poder executar outros.

Geralmente, são executados *scripts* adicionais e, ao invés de usar apenas uma ferramenta para realizar a busca por vulnerabilidades, são executadas uma suite de ferramentas. A ideia por trás disto, é utilizar as melhores características de cada ferramenta de forma a obter um resultado mais relevante em algumas áreas.

6.1.1 Ferramentas

Existem diversas ferramentas que podem automatizar a busca por vulnerabilidades em sistemas. Vamos apresentar um pouco das características de algumas delas a seguir.

OWASP Zed Attack Proxy

O *OWASP ZAP* é uma ferramenta open-source para busca de vulnerabilidades em aplicações web.

Algumas de suas funções incluem:

1. Proxy de Interceptação
2. Scanner ativo e passivo
3. Scanner por força-bruta
4. Spider
5. Fuzzer
6. Scanner de Portas
7. Ceritificados SSL dinâmicos
8. API
9. Integração com o Beanshell
10. Suporte para SmartCards

Algumas características do ZAP:

1. Fácil de instalar (requer Java)
2. Abrangentes páginas de ajuda
3. Totalmente internacionalizado
4. Em desenvolvimento ativo (em 2012)
5. Open-source
6. Multi-plataforma

Acompanhe na imagem 6.1 a tela de apresentação do ZAP.

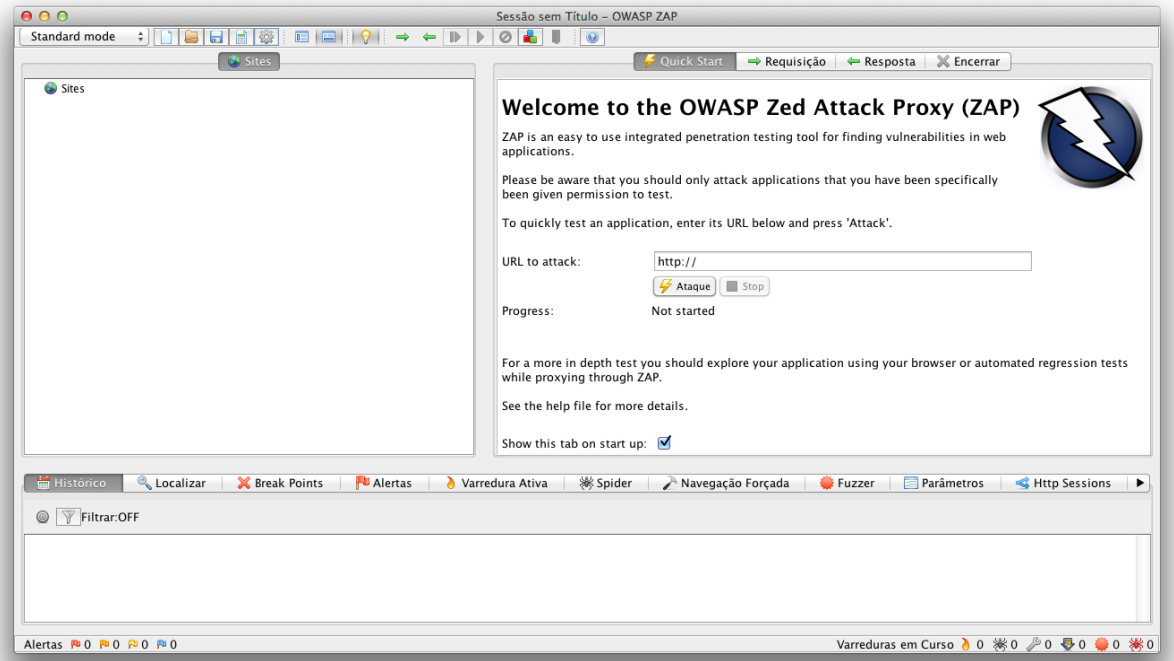


Figura 6.1: Tela de apresentação do OWASP ZAP

Burp Suite

Burp Suite é uma plataforma integrada para a realização de testes de segurança de aplicações web. Suas várias ferramentas trabalham em conjunto para apoiar o processo de teste completo, desde o mapeamento inicial e análise de superfície de um aplicativo de ataque, até encontrar e explorar vulnerabilidades de segurança.

O *Burp* lhe dá total controle, permitindo combinar avançadas técnicas manuais com o estado-da-arte da automação, para tornar seu trabalho mais rápido, mais eficaz, e mais divertido.

O *Burp Suite* contém os seguintes componentes-chave:

1. Um proxy de interceptação, que permite inspecionar e modificar o tráfego entre o navegador e o aplicativo de destino.

2. Spider, para rastreamento de conteúdo e funcionalidade.

3. Um scanner de aplicação web, para automatizar a detecção de vários tipos de vulnerabilidade.

4. Uma ferramenta Intruder, para a realização de poderosos ataques personalizados para encontrar e explorar vulnerabilidades incomuns.

5. Uma ferramenta Repeater, para manipular e reenviar pedidos individuais.
6. Uma ferramenta Sequencer, para testar a aleatoriedade de tokens de sessão.
7. A capacidade de salvar o seu trabalho e continuar a trabalhar mais tarde.
8. Extensibilidade, permitindo que você facilmente escrever seus próprios plugins, para executar tarefas complexas e altamente personalizada dentro *Burp Suite*.

Acompanhe na imagem 6.2 a tela de apresentação do Burp Suite.

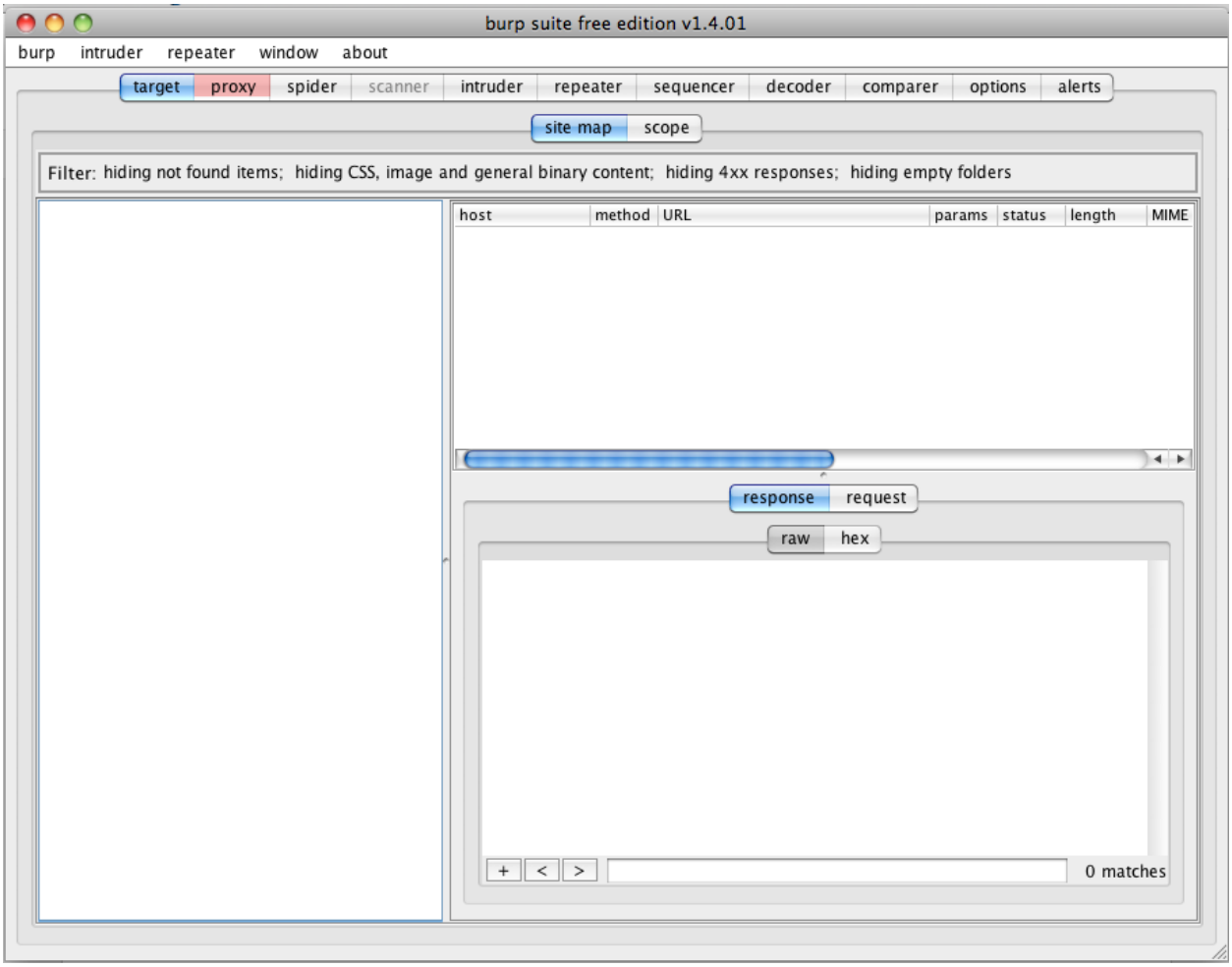


Figura 6.2: Tela de apresentação do Burp Suite

6.2 Tipos de Vulnerabilidades

A seguir, acompanharemos alguns tipos de vulnerabilidades que podem ser encontradas em aplicações web. Essas vulnerabilidades estarão categorizadas em grupos de acordo com

algumas de suas características, o que não significa que cada uma pertença a somente o grupo que está alocado nesta seção.

Essa seção está organizada fornecendo para cada categoria uma descrição e para cada vulnerabilidade uma breve explicação de sua ocorrência, com um exemplo e também com medidas para mitigá-las.

6.2.1 Exposição de Informação

Esta categoria é para vulnerabilidades que protegem dados sensíveis de maneira insegura. A proteção referida aqui inclui a confidencialidade e integridade dos dados durante seus ciclos de vida, incluindo o armazenamento e transmissão dos dados.

Note que esta categoria destina-se a ser diferente de problemas de controle de acesso, embora ambos falham para proteger os dados de forma apropriada. Normalmente, o objetivo do controle de acesso é a concessão de acesso a dados para alguns usuários, mas outros não. Nesta categoria, estamos em vez preocupado com a proteção de dados sensíveis que não se destinam a ser revelado ou modificados por qualquer usuário da aplicação. Exemplos deste tipo de dados sensíveis podem ser chaves de criptografia, senhas, tokens de segurança ou qualquer informação crítica de uma aplicação.

Comentários HTML e versionamento

Comentários em HTML ou informações de versionamento podem expor informações importantes sobre a aplicação, que, de certa maneira, podem acrescentar informações para viabilizar a descoberta de uma vulnerabilidade de maior gravidade.

Impacto As informações contidas em comentários HTML podem levar a descoberta de itens como:

- o fluxo de funcionamento da aplicação
- rotinas de verificação de conteúdo
- os métodos de controle de sessão
- observações sobre a codificação
- versões de frameworks de apoio utilizados

Exemplos Veja um exemplo na imagem 6.3.

Medidas Inspecione o código da aplicação e dos templates buscando por comentários que possam ser impressos no código HTML. É interessante buscar também por possíveis comentários em funções de debug ou que venham a descrever erros.

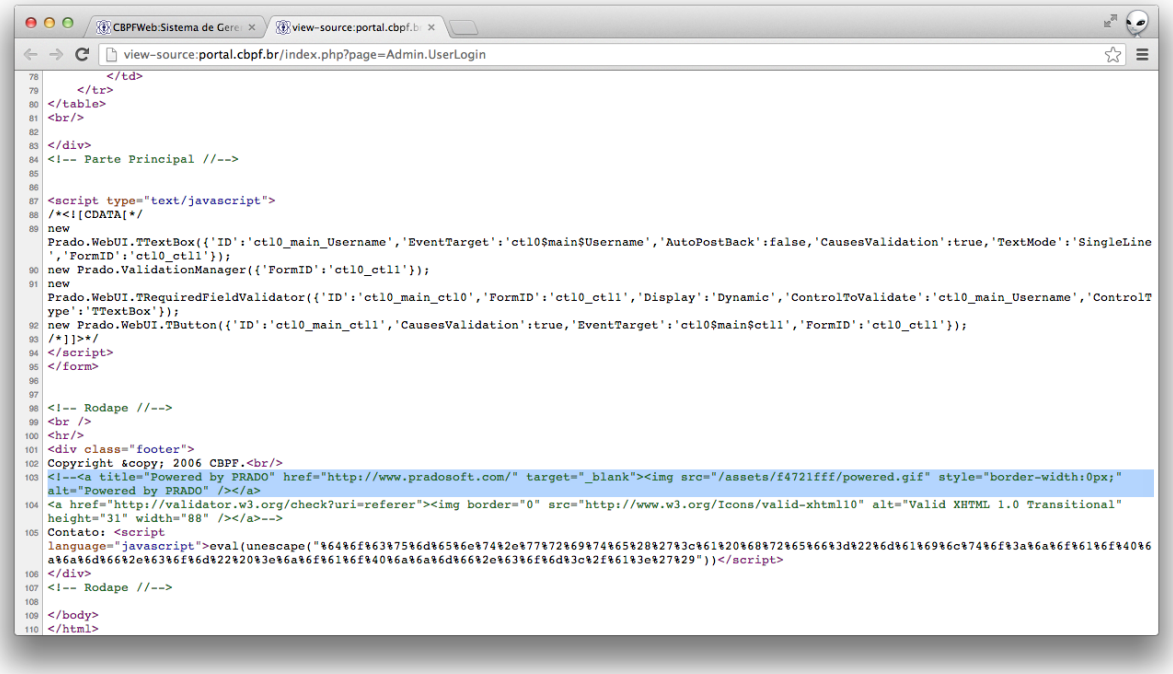


Figura 6.3: Comentários HTML e versionamento

Base de dados locais

Base de dados locais podem expor completamente os dados armazenados na aplicação web. Embora seja muito comum o uso de bancos *SQLite* devido sua simplicidade de implementação e manutenção, muitas vezes a proteção dos recursos do banco não são tão bem protegidos.

Impacto Um atacante com posse de uma base de dados pode, caso tenha sido encriptada, realizar ataques de força bruta localmente com a finalidade de descobrir os dados.

Caso consiga acesso a uma base sem criptografia, ou consiga descriptografar uma, ele poderá fazer uso indiscriminado dos dados nela, podendo assim, usar os dados para cometer grande parte dos tipos de incidentes possíveis, como:

- Roubo de informações pessoais, que podem ser usadas para compras, por exemplo
- Acesso a informações de configuração da aplicação
- Acesso a estrutura interna da aplicação, entre outros acessos.

Medidas Realizar a proteção dos arquivos de base de dados através da configuração do servidor web é uma das possíveis medidas. O uso de bases locais é desencorajado, devido, além dos problemas de segurança, a problemas de performance.

Mensagens de Erro e de Exceção

Essa vulnerabilidade ocorre quando a aplicação web não realiza o tratamento adequado das mensagens de erro geradas e, consequentemente, acaba expondo informações da tecnologia e do ambiente usado no sistema. Analisando as informações retornadas, é possível identificar falhas na aplicação e mapear a estrutura interna da aplicação, por exemplo.

Impacto Diversas mensagens de erro podem ser expostas ao usuário de forma detalhada. Dependendo da mensagem de erro, podem ser obtidos:

- erros da aplicação
- erros em requisições da aplicação
- erros de componentes da aplicação
- informações sobre a arquitetura da aplicação

Exemplos Veja um exemplo na imagem 6.4.

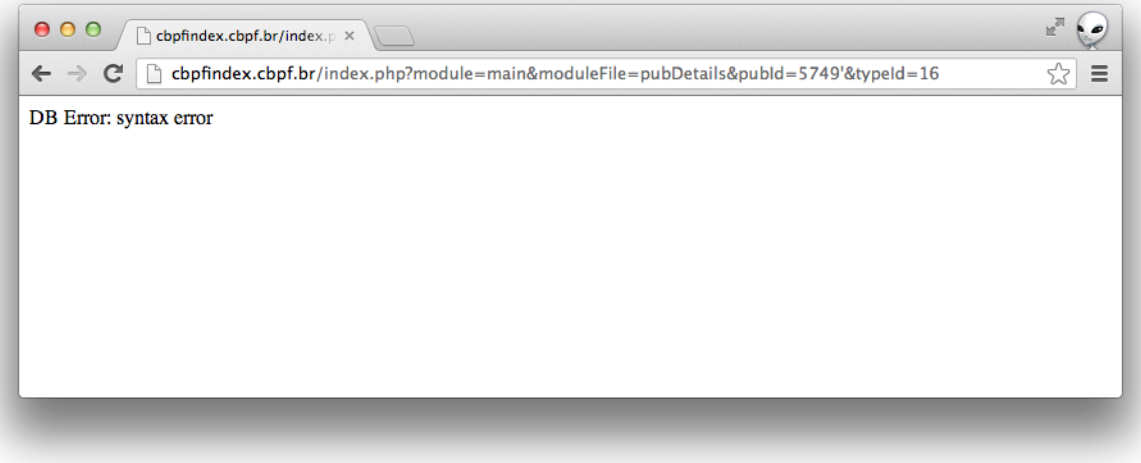


Figura 6.4: Mensagens de Erro e de Exceção

Medidas Recomenda-se remover as mensagens de erro da aplicação e o detalhamento de erros. Isso pode ser feito, adicionando uma mensagem de erro genérica ou retornar a aplicação para a sua página raiz.

Indexação de Diretório e Leitura de Arquivos Locais (Local File Read)

Essa vulnerabilidade permite que diretórios e arquivos internos da aplicação fiquem acessíveis externamente.

Impacto Esses arquivos internos da aplicação podem revelar informações internas do serviço ou do ambiente que podem ser utilizadas para a elaboração de ataques mais sofisticados. Além disso, informações internas sobre o funcionamento da aplicação web podem ser descobertas ou divulgadas fora do previsto.

Exemplos Veja um exemplo na imagem 6.5.

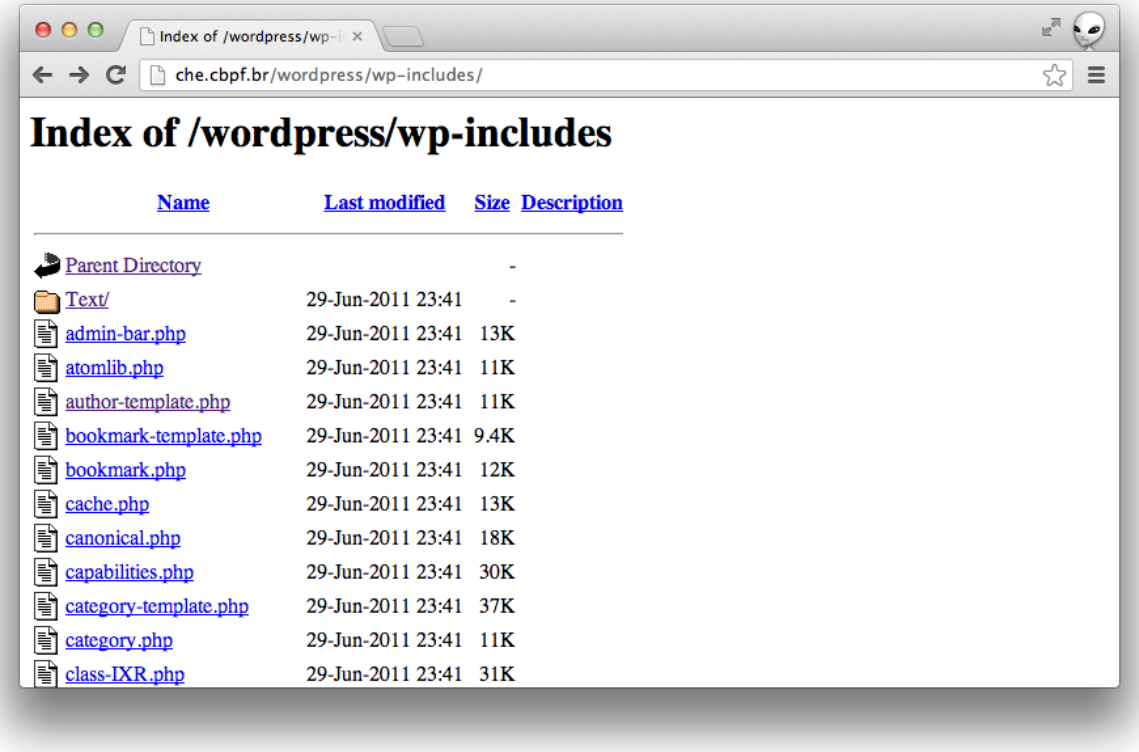


Figura 6.5: Indexação de Diretório

Medidas O acesso a recursos desnecessários do ambiente deve ser bloqueado. A listagem de diretórios deve ser desativada no servidor web e seções com informações internas da aplicação devem estar restritas.

Descoberta de Path

A vulnerabilidade de Descoberta de Path ou Descoberta de Caminho permite a um indivíduo mal-intencionado saber em qual diretório se encontram os arquivos da aplicação Web. Algumas vulnerabilidades, como as que usam a função *load_file()* podem permitir a visualização do código da página, mas para isso o atacante tem que conhecer o caminho completo do recurso que deseja acessar.

Mais informações em podem ser obtidas em: `https://www.owasp.org/index.php/Full_Path_Disclosure` e uma ferramenta para descoberta de path pode ser obtida em: `https://code.google.com/p/inspathx/`.

Impacto Atacantes podem abusar do conhecimento, por exemplo da raiz onde se encontra o diretório web, e usá-lo em combinação com outras vulnerabilidades, como Leitura de Arquivos Locais para roubar arquivos de configuração da aplicação web ou do resto do sistema operacional.

Medidas Recomenda-se remover as mensagens de erro da aplicação e o detalhamento de erros. Isso pode ser feito, adicionando uma mensagem de erro genérica ou retornar a aplicação para a sua página raiz.

Recomenda-se também o uso de bibliotecas seguras, que, por padrão, não imprimam mensagens e sim retornem códigos de erro.

Inclusão de Arquivos Remotos (Remote File Inclusion)

A Inclusão de Arquivos Remotos é uma das vulnerabilidades mais procuradas nos servidores web atualmente. Isso ocorre pois através dele, um indivíduo mal-intencionado pode:

- Realizar execução de código no servidor web
- Executar código client-side como um JavaScript, que pode levar a outros ataques, como XSS
- Causar Negação de Serviço (DoS)
- Pode realizar manipulações nas informações fornecidas

Impacto Além de expor completamente informações e conteúdo da aplicação, uma vulnerabilidade de Inclusão de Arquivos Remota pode expor também todos os serviços executados no servidor por permitir a inclusão de um arquivo que pode ser um shell via web[1].

Medidas Algumas medidas como:

- Nunca use includes baseados em informações fornecidas pelo usuário. Quando for o caso, não use `if/elseif/else` use `switch/case`.
- Independente dos dados, faça sempre trimming das strings de `include`, procurando por `http, /, //`
- Desabilite as opções *registrer_ global, allow_ url_ fopen e allow_ url_ include* no arquivo php.ini
- Valide fortemente os dados fornecidos pelos usuários

[1] Um exemplo de shell pode ser encontrado em `http://laudanum.inguardians.com/`

Path Traversal e Null Bytes

A vulnerabilidade de *Path Traversal* permite ao atacante acessar arquivos e diretórios que são armazenados fora da pasta raiz da web. Ao navegar no aplicativo, o atacante procura por links absolutos para arquivos armazenados no servidor web. Assim, eles manipulam o valor das variáveis para arquivos de referência com 'ponto-ponto-barra (../)"sequências e suas variações, verificando se é possível acessar arquivos arbitrários e diretórios armazenados no sistema de arquivos.

Impacto Um atacante explorando esse ataque pode ter acesso a informações que podem elevar seu privilégio e comprometer toda a aplicação.

Os dados que podem ser obtidos são:

- código fonte do aplicativo
- configuração
- arquivos do sistema

Este ataque pode ser executado com um código externo malicioso injetado no caminho, como um ataque de injeção de recursos. Para executar este ataque não é necessário o uso de uma ferramenta específica; atacantes costumam usar um rastreador/spider para detectar todas as URLs disponíveis.

Exemplos

```
http://some\_site.com.br/get-files.jsp?file=report.pdf
http://some\_site.com.br/get-page.php?home=aaa.html
http://some\_site.com.br/../../../../etc/shadow
http://some\_site.com.br/get-files?file=/etc/passwd
```

Medidas Algumas medidas como:

- Independente dos dados, faça sempre trimming das strings de `include`, procurando por `http, /, //`
- Desabilite as opções *registrer_ global, allow_ url_ fopen e allow_ url_ include* no arquivo php.ini
- Valide fortemente os dados fornecidos pelos usuários
- Veja se existe a possibilidade de executar o servidor web em chroot[2]

[2]Exemplo de implementação em: `http://www.debian.org/doc/manuals/securing-debian-howto/ap-chroot-apache-env.en.html`

Injeções

Falhas de injeção permitem que atacantes insiram código malicioso através de uma aplicação web para outro sistema. Estes ataques incluem chamadas para o sistema operacional através de chamadas do sistema, e também no uso de programas externos através de comandos de shell, bem como chamadas para back-ends de bancos de dados através de SQL (ou seja, injeção de SQL). Geralmente scripts escritos em PHP, Perl, Python e outras linguagens podem ser vulneráveis a conteúdo injetado quando usados em aplicações web mal concebidas ou executadas. Toda vez que uma aplicação Web usa uma linguagem interpretada de qualquer tipo existe o perigo de um ataque de injeção.

Muitas aplicações Web usam recursos do sistema operacional e programas externos para executar suas funções. O *Sendmail* é provavelmente o programa mais frequentemente invocado externamente, mas muitos outros programas são usados também. Quando uma aplicação web passa informações a partir de uma solicitação HTTP através como parte de um pedido externo, este deve ser cuidadosamente validado e limpo. Caso contrário, um atacante pode injetar (meta)-caracteres ou caracteres especiais, comandos maliciosos, ou modificadores de comando para que a aplicação web, cegamente, transmita-os para o sistema externo para execução.

A Injeção de SQL é uma forma particularmente comum e perigosa de injeção. Para explorar uma falha de injeção SQL, o atacante deve encontrar um parâmetro que a aplicação web passa através de um banco de dados. Com cuidado, um atacante pode incorporar comandos SQL maliciosos no conteúdo do parâmetro, assim, o atacante pode enganar o aplicativo web e encaminhar uma consulta maliciosa ao banco de dados.

Impacto Ataques de injeção podem ser descobertos e explorados de forma fácil, mas também podem ser extremamente obscuros. As consequências do ataque podem pertencer a uma gama de gravidade, do trivial até o comprometimento do sistema ou sua destruição. Em qualquer caso, o uso de chamadas externas é bastante difundida, por isso a probabilidade de uma aplicação web com uma falha de injeção deve ser considerada elevada.

Exemplos Veja um exemplo na imagem 6.6.

Medidas Para contornar esse problema, a principal recomendação é realizar uma validação de todos os dados de entrada, principalmente no lado do servidor, a fim de evitar a utilização de caracteres maliciosos. A validação de dados realizada somente no lado do cliente pode ser facilmente contornada copiando a página e removendo o código de validação, geralmente feito em javascript[3].

Para realizar a filtragem dos dados recomenda-se utilizar uma abordagem positiva, que consiste em negar todas as entradas com exceção dos dados previamente escolhidos. Por exemplo, não há necessidade de um campo do tipo data no formulário receber entradas com valores diferentes de números e talvez barras '/' (para separar dia, mês e ano). Dessa forma,

[3]Informações em: `http://pt.wikipedia.org/wiki/JavaScript`

Figura 6.6: Injeções

o desenvolvedor não terá que adivinhar ou atualizar todas as formas de entradas maliciosas para negar todos os caracteres específicos.

6.2.2 Configurações e Manutenção

Arquivos de configuração

A vulnerabilidade relacionada a arquivos de configuração se concentra principalmente nas informações fornecidas por eles e o quanto isso pode impactar nos ativos da organização caso alguma brecha de segurança possa ser encontrada devido o não cumprimento de uma boa prática de configuração segura.

Impacto Informações que podem ser encontradas em arquivos de configuração, podem dizer respeito a:

- informações para acesso a serviços externos (banco de dados, serviços de log, etc)
- comentários sobre configurações
- versões de software, como última atualização ou referências externas

Medidas O cumprimento de boas práticas de segurança e recomendações de configuração feitas pelo fabricante é desejado. É necessário verificar sempre se os arquivos de configuração da aplicação estão protegidos contra acesso externo e também acesso interno de recursos não autorizados da prórpia aplicação.

Arquivos de Backup ou de versões antigas

Enquanto a maioria dos arquivos em um servidor web são manuseados diretamente pelo próprio servidor, não é incomum encontrar arquivos sem referência e/ou esquecidos que podem ser usados para obter informações importantes sobre qualquer infra-estrutura ou as credenciais.

Os cenários mais comuns incluem a presença de versões antigas renomeadas de arquivos modificados, arquivos de inclusão que são carregados para a língua de escolha e pode ser baixado como fonte, ou até mesmo backups automáticos ou manuais em forma de arquivos compactados. Os arquivos de backup também podem ser gerados automaticamente pelo sistema de arquivos e estarem junto dos arquivos originais.

Impacto Todos esses arquivos podem conceder o acesso a um indivíduo mal-intencionado as funcionalidades internas da aplicação, backdoors, interfaces administrativas, ou até mesmo as credenciais para se conectar à interface administrativa ou o servidor de banco de dados.

Medidas Para garantir uma estratégia de proteção eficaz, o teste deve ser agravado por uma política de segurança que proíbe claramente práticas perigosas, tais como:

- Edição local dos arquivos da aplicação web. Este é um hábito ruim, uma vez que é provável que se gerem arquivos de backup dos editores. É incrível ver como muitas vezes isso é feito, mesmo em grandes organizações. Se você precisa editar arquivos em um sistema de produção, você precisa garantir que não está deixando nada que é absolutamente previsto para trás.

- Verifique cuidadosamente qualquer outra atividade realizada em sistemas de arquivos expostos pelo servidor web, tais como atividades de administração local. Por exemplo, se você ocasionalmente precisa tirar um snapshot de alguns diretórios (que você não deve, em um sistema de produção ...), você pode pensar em compactá-los primeiro. Tenha cuidado para não esquecer de retirar esses arquivos compactados!

- Políticas de gestão e de configuração devem ajudar a não deixar soltos arquivos obsoletos e sem referência.

- Arquivos de dados, arquivos de log, arquivos de configuração, etc, devem ser armazenadas em diretórios não acessíveis pelo servidor web, para contrariar a possibilidade de divulgação de informações (para não mencionar a modificação de dados se as permissões de diretório web permitem escrever ...).

- Instantâneos do sistema de arquivos não devem ser acessíveis via web, se a pasta raiz da sua configuração do servidor web não permitir isso, configure o servidor web para negar o acesso a esses arquivos, por exemplo.

Acesso Externo à Interface de Administração

Interfaces administrativas são um dos poucos controles dentro do Guia de Desenvolvimento, que é legalmente obrigatória (dentro dos EUA). Para organizações que desejam seguir as normas SOX e ISO 17799, elas exigem que funções administrativas sejam separadas das funcionalidades normais, pois é um controle chave de fraudes.

Ao projetar aplicações, é necessário mapear as funcionalidade administrativas e assegurar que os controles de acesso e de auditoria são apropriados e estão em seu devido lugar.

Essa vulnerabilidade depende da política de segurança implementada, mas é uma boa prática manter a interface de administração sempre de maneira bem restrita.

Impacto O acesso não controlado a interface de administração pode expor a aplicação a diversos tipos de ataques, como ataques de força bruta. Tentativas de exploração visando contornar os mecanismos de autenticação, entre outros riscos.

Exemplos Veja um exemplo na imagem 6.7.

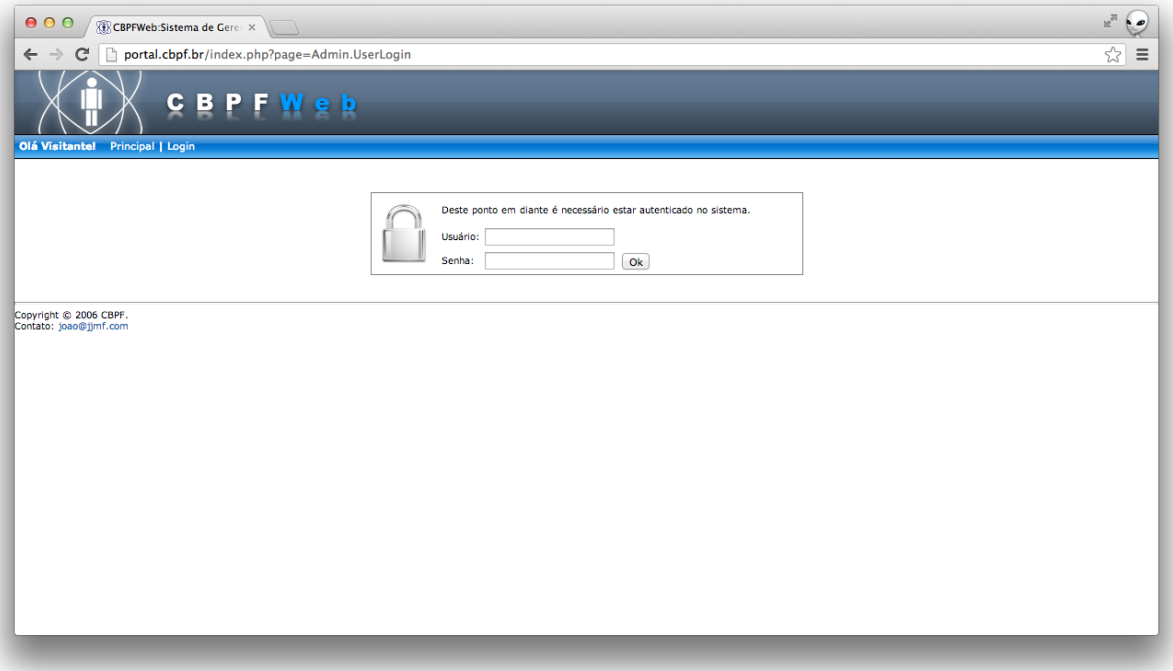


Figura 6.7: Acesso Externo à Interface de Administração

Medidas Recomenda-se que a interface de administração da aplicação web seja segregada e com acesso restrito somente aos administradores da aplicação. Usuários da aplicação não são administradores logo o acesso deles a interface de administração não é necessário.

Quebra de Autenticação e Roubo de Sessão

Gerenciamento de autenticação e de sessão inclui todos os aspectos da manipulação de autenticação de usuários e gerenciamento de sessões ativas.

O Gerenciamento de autenticação é um dos fatores críticos de uma aplicação, em conjunto com seus mecanismos, como, por exemplo, o 'Esqueci minha senha" ou "Lembre-me minha senha" e os mecanismos para alteração de senhas.

Ataques aos mecanismos de autenticação em aplicações web são previstos e é uma boa prática que todas as funções de gerenciamento de contas devem requerer re-autenticação, mesmo na presença de outros mecanismos de segurança.

As aplicações web também devem estabelecer sessões para acompanhar o fluxo de pedidos de cada usuário. Como o HTTP não fornece esse recurso, a aplicação deve se encarregar dele, e também deve fornecer meios de controlar o fluxo de requisições de seus usuários. Uma alternativa é a utilização de tokens de sessão, mas muitos desesnvolvedores usam métodos próprios que permitemou que não sejam fortes o suficientes, de forma a manter a aplicação vulnerável.

A menos que todas as credenciais de autenticação e identificadores de sessão sejam protegidos com SSL em todas as vezes, e também protegidos contra a divulgação de outras falhas, tais como cross site scripting, um atacante pode seqüestrar uma sessão do usuário e assumir sua identidade.

Impacto Tais falhas podem permitir que algumas ou até mesmo todas as contas possam ser atacadas. Uma vez bem sucedido, o atacante detem todas as credenciais e premissões de sua vítima. Contas privilegiadas são freqüentemente alvo desses ataques.

Medidas Para a proteção do ambiente, é necessário tomar algumas medidas como:

- o uso de senhas fortes pelos usuários
- a definição de uma política de senha, que possua a capacidade de bloquear o login do usuário depois de certo número de conexões falhas
- mecanismos de controle de mudanças, onde as funções relacionadas com o gerenciamento de contas sempre exijam a re-autenticação do usuário
- armazenamento seguro das senhas, onde as senhas serão armazenadas com o uso de um algoritmo de hash criptográfico forte
- proteção das credenciais em transito, onde as credenciais utilizadas são enviadas para o usuário de forma segura
- proteção do session ID ou do Identificador de Sessão de modo que, se alguma outra pessoa não autorizado tiver posse dele, essa pessoa não consiga asssumir a identidade de outra pessoa

- não permita a enumeração das contas disponíveis
- remova todos os códigos de demonstração disponíveis
- elabore trinhas de auditoria e logs compretos do uso administrativo da aplicação

HTTP Response Splitting

A vulnerabilidade conhecida como *HTTP Response Splitting* é proveniente de uma falha da aplicação ou do seu ambiente quando ele não higieniza adequadamente os valores de entrada.

O ataque consiste em fazer o servidor imprimir um carriage return (CR, ASCII 0x0D) e um avanço de linha (LF, ASCII 0x0A) seguido do conteúdo fornecido pelo atacante na seção de cabeçalho da sua resposta, normalmente, incluindo-os em campos de entrada enviados para a aplicação. De acordo com a norma HTTP (RFC 2616), os cabeçalhos são separados por um CRLF e os cabeçalhos da resposta são separados a partir do seu corpo por dois. Portanto, falhando ao remover CRs e LFs, a aplicação permite que o atacante defina cabeçalhos arbitrários, assuma o controle do corpo, ou quebre a resposta em duas ou mais.

Impacto Um atacante em posse de uma vulnerabilidade de HTTP Response Splitting obtem praticamente o controle total do que será exibido para o cliente.

Os ataques que podem ser efetuados são:

- Cross-site scripting (XSS)
- Defacement
- Envenenamento de cache
- Hijacking, entre outros semelhantes

Medidas A solução genérica é codificar strings na URL antes da inclusão no cabeçalhos HTTP, como nos campos *Location* ou *Set-Cookie*.

Exemplos típicos de sanitização incluem a conversão para inteiros ou substituição usando expressões regulares. Apesar da divisão da resposta não ser específica para PHP, o interpretador PHP contém proteção contra o ataque desde a versão 4.4.2 e 5.1.2.

6.2.3 Validação de Dados

Vulnerabilidades relacionadas com a Validação de Dados costumam ser as mais presentes em aplicações, mesmo em aplicações desktop.

Embora sejam as mais comuns, elas representam um grande risco para a aplicação. Primeiro, porque permitem a manipulação da informação presente na aplicação e segundo, porque podem facilmente denegrir a imagem da organização dona da aplicação.

Cross-Site Scripting (XSS)

Ataques de *Cross-Site Scripting* (XSS) ocorrem quando scripts maliciosos são injetados dentro do contexto de seções aparentemente confiáveis de um site. Um atacante pode fazer uso da aplicação para disseminar um código malicioso, geralmente na forma de um script que é executado no navegador do usuário final. Essa falha ocorre geralmente em aplicações Web que não validam, interpretando as entradas que partem do lado do cliente sem o devido tratamento. Qualquer funcionalidade ou recurso disponível para o navegador pode ser explorada no ataque de XSS (códigos HTML, JavaScript, Flash, applets Java).

Um *XSS Reflected* ocorre quando o código injetado é passado e refletido pelo Servidor Web durante uma mensagem de erro, resultados de buscas ou quaisquer outra resposta que inclui, parcial ou integralmente, os parâmetros passados ao servidor.

Um *Stored XSS* ocorre quando o código injetado é armazenado persistentemente no lado do servidor, dentro de um banco, fórum de mensagens, campo de comentários, e etc. A vítima recebe os scripts maliciosos do servidor sempre que requisitar um endereço carregando as informações armazenadas.

É importante observar que, apesar de vários navegadores (Chrome e IE, por exemplo) já incluírem filtros nativos contra *Reflected Cross Site Scripting*, existem técnicas conhecidas para contornar essas proteções. No caso de um *Stored Cross Site Scripting*, o navegador não consegue discernir o conteúdo original do conteúdo injetado, carregando automaticamente as informações.

Impacto Um atacante pode usar um XSS e enviar uma URL maliciosa para um usuário final. O navegador recebe os parâmetros e executa automaticamente os scripts, que podem acessar cookies, sequestrar seções e reter informações sensíveis. Os scripts podem, inclusive, manipular e modificar o conteúdo da página HTML, exibindo um conteúdo arbitrário. É possível criar frames maliciosos dentro do contexto original da página, manipular notícias, controlar o navegador do usuário, entre outros.

Exemplos Veja um exemplo na imagem 6.8.

Medidas Esta classe de vulnerabilidade pode ser evitada através de três medidas principais:

- Todo parâmetro recebido pela aplicação deve ser validado quanto ao tipo e tamanho para verificar se a entrada é compatível com o dado esperado. Recomenda-se a utilização de uma abordagem "Whitelist"ao invés de "Blacklist- aceitar apenas caracteres compatíveis com o tipo de dado esperado. Todo tipo de entrada de dado inválida deve ser rejeitada, ao invés de ser tratada e não deve ser retornada em mensagens de erro da aplicação.

- Todo dado entrado deve sofrer codificação 'HTML Encode"antes de ser copiado para respostas da aplicação. Todos os meta-caracteres HTML, incluindo $< > "'$ e $=$, devem

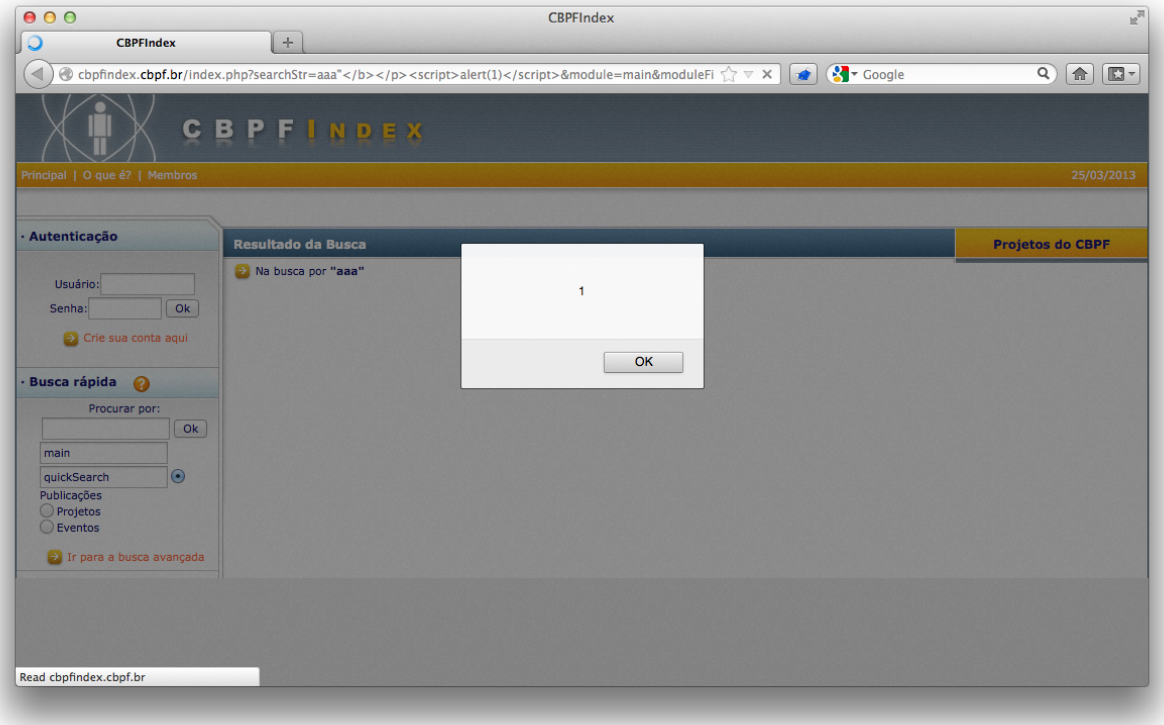


Figura 6.8: Cross-Site Scripting (XSS)

ser substituídos pelas correspondentes entidades HTML (< > etc). Se o dado for retornado dentro de código JavaScript deve-se utilizar o formato \HH para codificar os caracteres especiais.

- É importante que todo o código da aplicação seja revisado para proteger a aplicação contra falhas de XSS em outras URLs e páginas. A expressão regular de validação de e-mail deve ser revalidada em todos os campos em que é usada. Um controle de tamanho das entradas e formulários também é útil para mitigar e dificultar a elaboração de ataques.

Caso a aplicação tenha funcionalidade que permita a inclusão de algumas tags HTML (o que deve ser evitado), a aplicação deve realizar uma validação rigorosa para garantir que apenas algumas tags autorizadas são permitidas e que nenhuma tag perigosa é aceita. Esta validação deve ser realizada através de expressões regulares.

A OWASP ESAPI[4] é uma API consolidada que agrega filtros e controles para tratamento de problemas desse tipo para várias linguagens.

[4]Mais informações disponíveis em: `https://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API`

Para maiores informações sobre como prevenir XSS consulte: `https://www.owasp.org/index.php/XSS_%28Cross_Site_Scripting%29_Prevention_Cheat_Sheet`

Comment Spam

Esses problemas podem ser relativos à programação das seções de comentários ou pode estar relacionado à uma vulnerabilidade qualquer do site. Os comentários de Spam são inseridos de maneira automatizada com o uso de scripts e robôs que varrem a Internet e inserem textos em páginas vulneráveis ou com seções de comentários abertas. Isso é mais comum em seções de comentários, mas também pode acontecer em páginas convencionais, inserindo o spam na própria página HTML (usando campos hidden por exemplo).

O método é usado por *spammers* para abusar da reputação e do ranking de SEO da página original, fazendo com que seu site ganhe maior relevância nas pesquisas. A maior parte desse *comment spam* está relacionado a produtos farmacêuticos (*buy viagra, buy cialis* etc) e pornografia.

Impacto Conteúdo de Spam em comentários podem denegrir fortemente a imagem de um website. Eles são ruins pois interferem na interface do usuário de maneira agressiva, ocupam espaço e prejudicam a classificação em mecanismos de busca.

De forma resumida, a vulnerabilidade de *comment spam* permite:

- diminuir a qualidade do site
- reduzir a qualidade da experiência do usuário
- afetar a disponibilidade da aplicação web, pois em alguns casos podem ser executados ataques automatizados com a finalidade de deixar a aplicação mais lenta, ou de comprometer o mecanismo de armazenamento
- afetar o posicionamento do website em mecanismos de busca

Medidas Algumas medidas para evitar esse tipo de problema:

- Uso de CAPTCHA's, exigindo uma confirmação visual para verificar se um humano está digitando o comentário. É claro que isso não impede que usuários insiram comentários de spam manualmente, mas já reduz a quantidade de mensagens inseridas.
- Existem módulos que verificam o conteúdo da mensagem de comentário, mas isso não é eficiente já que os spammers escondem as URL's com mensagens aparentemente normais comentando o artigo (como no caso da revista).
- Assegurar que o site não está comprometido. Quando o site está vulnerável, scripts automáticos podem explorar falhas e acessar seções administrativas, inserindo mensagens e auto-aprovando os comentários, por exemplo.

- Proibir a inserção de URL's nos campos de comentários (incluindo nas informações de usuário).

Em todos os casos, é importante verificar e moderar o conteúdo das seções interativas, removendo estas mensagens de spam. Caso as mensagens persistam, os mecanismos de busca podem identificar e diminuir a confiança e o ranking SEO do domínio original.

Redirecionamento sem Validação

Uma vulnerabilidade de Redirecionamento sem Validação, ou *Open Redirect* ocorre quando a aplicação web recebe um parâmetro e redireciona o usuário sem qualquer tipo de validação. Um parâmetro HTTP pode conter um valor que permite à aplicação redirecionar a requisição para uma URL específica. A aplicação recebe a entrada e usa esse valor como destino do redirecionamento.

Caso o redirecionamento ocorra sem o devido tratamento e validação, qualquer usuário pode abusar da aplicação para criar links maliciosos para campanhas de phishing ou páginas falsas, por exemplo.

Impacto Como o nome e domínio do servidor é o mesmo do site original, os ataques ganham maior credibilidade, conseguindo comprometer uma número maior de usuários. Estes ataques podem ser usados para roubar dados, informações e credenciais de usuários, por exemplo.

Medidas Para prover um redirecionamento seguro, é interessante adotar as seguintes práticas:

- Não usar dados provenientes do cliente
- Usar uma codificação interna da aplicação para criptografar a URL e evitar que o usuário consiga manipular os parâmetros. Dessa forma, somente quem conhece a chave interna da aplicação conseguiria descriptografar o conteúdo da aplicação.
- Verificar se o usuário pode acessar o recurso que está tentando.

Cross-Site Request Forgery (CSRF)

O *Cross-Site Request Forgery* (CSRF) é uma classe de ataques que explora a relação de confiança entre uma aplicação Web e seu usuário legítimo. Para execução do CSRF, o usuário mal intencionado deve induzir o utilizador legítimo, seja por meio de engenharia social ou outros artifícios como *Cross-Site Scripting*, a executar atividades arbitrárias no aplicativo, permitindo que virtualmente qualquer ação específica possa ser realizada no sistema sem o consentimento do usuário final.

A exploração do CSRF se dá por meio de aplicativos vulneráveis que não possuem *tokens*/controles específicos para garantir que todas as comunicações com o usuário legítimo

sejam feitas dentro de um mesmo contexto, ou seja, utilizando um fluxo lógico de execução. Um bom exemplo seria a execução de uma transação de movimentação financeira de créditos para uma nova titularidade (usuário), sem que a tela para captura dos dados de origem e destino seja solicitada pelo usuário (vítima do ataque).

O alvo do atacante geralmente são as transações valiosas e mais comuns nos sistemas de aplicativos web, tais como:

- Alteração de e-mail ou dados pessoais;
- Alteração de senha de acesso;
- Movimentações financeiras ou compras online.

Aplicações vulneráveis são exploradas basicamente por meio das seguintes etapas:

- O usuário se autentica ou está autenticado na aplicação alvo do ataque;
- O usuário recebe um link ou utiliza o mesmo navegador para acessar um aplicativo malicioso;
- O link ou aplicativo malicioso navegado inclui uma requisição à aplicação alvo, carregando todos os parâmetros necessários para a execução da transação;
- Pela *SameOriginPolicy* do navegador, como existe uma sessão autenticada válida para o usuário no aplicativo alvo, a aplicação recebe a requisição e executa a operação conforme a solicitação enviada.

Impacto Com uma aplicação web vulnerável a CSRF é possível através de outra aplicação web realizar requisições devido a existência de uma sessão sem controle na aplicações vulnerável a CSRF. Isso ocorre porque como não existe um controle de sessão, as requisições serão sempre estáticas, com os mesmos parâmetros e valores, logo será sempre possível repeti-las ao longo do tempo.

Para realizar um teste é necessário criar um código em outra página, externa a aplicação inicial, que irá fazer uma requisição a aplicação vulnerável sem nenhuma interação com o usuário.

Sabendo que o navegador inclui automaticamente nas solicitações para um mesmo aplicativo todas e quaisquer credenciais associadas a ele, seja um cookie de sessão do usuário, token e credenciais de autenticação e como a aplicação não avalia o fluxo de execução das operações, ela não tem condições de distinguir a solicitação maliciosa da solicitação legítima. Dessa forma, ao fazer a requisição, o navegador irá possuir as credenciais necessárias, logo a requisição será processada e executada, pois nenhum controle para verificar a origem da requisição foi feito.

Dentre as formas mais comuns que o atacante pode utilizar para explorar o CSRF em aplicações vulneráveis, destacam-se:

- Por meio de tags HTML
- Uso de IMG SRC `<img src='http://host/?command'>`
- Uso de SCRIPT SRC `<script src='http://host/?command'>`
- Uso de IFRAME SRC `<iframe src='http://host/?command'>`
- Por meio de objeto *XmlHttpRequest* (XMLHTTP).
- Por meio de código Javascript

Medidas Uma solução comum adotada pelos desenvolvedores contra o CSRF é o *Synchronizer Token Pattern*, que gera um token de "desafio" aleatório associado com a sessão atual do usuário. Esses tokens são inseridos nos formulários HTML, links associados com operações sensiveis e incluídas nas solicitações HTTP quando enviadas para o aplicativo web.

Quando o usuário gera uma solicitação por meio de um formulário, um campo 'input" do tipo 'hidden" deverá conter o token de proteção CSRF. Toda vez que uma transação sensível for executada, o aplicativo irá comparar o token enviado com o valor armazenado em sessão. Caso positivo, a transação será realizada e um novo token gerado de maneira aleatório. Em caso negativo, o aplicativo deverá recusar a execução da transação.

Uma requisição externa não consegue tomar conhecimento deste token identificador sem interagir com a página. É importante usar valores complexos (que não podem ser facilmente descobertos por força bruta) e corrigir falhas de *Cross-Site-Scripting*, que podem ser abusadas para encontrar o token e forçar requisições maliciosas.

Além disso, é importante proteger todos os campos e formulários interativos, revisando todo o fluxo das telas para evitar a exposição dos usuários do aplicativo.

A OWASP possui um framework de prevenção contra ataques de CSRF específico para Java, mas com espeficicações para outras linguagens, que pode ser acessado em: `https://www.owasp.org/index.php/Category:OWASP_CSRFGuard_Project`

Referência insegura à objetos

Uma referência insegura à objeto ocorre quando o desenvolvedor expõe uma referência a objeto usado internamente, como um arquivo, diretório, dados de banco de dados, uma chave primária, ID's, dentre outros. Um atacante pode manipular essas referências à objetos para acessar outros objetos sem autorização, caso não existia um controle de acesso.

Impacto Através da manipulação dos valores nos parâmetros passados via requisições GET, geralmente é possível encontrar essa vulnerabilidade. Não somente encontrar, mas dependendo do objeto que está sendo manipulado, é possível explorar outras vulnerabilidades, como, por exemplo:

- Leitura de Arquivos Locais

- Path Transversal

Algumas medidas geralmente implantadas para uma suposta proteção da aplicação é o uso no código (hard-coded) da extensão do objeto que será retornado. Uma forma de contornar essa restrição, é através da adição do carácter fim de linha - %0000 após o novo valor para o parâmetro.

Exemplos

Medidas Não se deve confiar nas informações passadas pelo usuário, restringindo e limitando o acesso do sistema aos componentes realmente utilizados para seu funcionamento. A aplicação deve verificar se o usuário está autorizado a exibir o que foi solicitado. O acesso a arquivos deve ser bloqueado, limitando a visualização para os objetos pertinentes.

6.2.4 Disponibilidade

Slow HTTP

Essa vulnerabilidade está relacionada a configuração do servidor web e ao contrário de outros ataques que podem comprometer a disponibilidade da aplicação, ele não se caracteriza pelo envio de grande quantidades de informações de forma a exaurir os recursos para a aplicação.

A exploração da vulnerabilidade Slow HTTP é feita através através do esgotamento das conexões possíveis do servidor web com o uso de conexões lentas, deixando as demais sem resposta.

Impacto O ataque de *Slow Headers* ocorre quando um atacante mantém as conexões do servidor abertas enviando requisições HTTP parciais, submetendo os cabeçalhos subsequentes em um intervalo de tempo que impossibilita o servidor de fechar os sockets. Desta maneira, o servidor web fica indisponível porque o número de sockets disponíveis diminui, aumentando o uso de memória (principalmente se o servidor aloca uma thread para cada conexão).

Uma das razões para esse comportamento é que alguns servidores têm temporizadores de "no data", que são reiniciados toda vez que algum byte chega no socket. O grande problema é que para grande parte dos servidores web não existe um tempo limite para a requisição como um todo. Um atacante pode, desta maneira, enviar um byte da requisição por vez durante vários minutos, ao invés de enviar toda a requisição em um único pacote, mantendo a conexão "presa" por tempo indeterminado.

Ataques de *Slow HTTP POST* acontecem quando o atacante mantém as conexões abertas enviando cabeçalhos de HTTP POST codificados com um campo de 'Content- Length" legítimo, informando a quantidade de dados esperada ao servidor web. Depois de concluir o envio dos cabeçalhos HTTP POST, o corpo da mensagem é enviado em longos intervalos de tempo para prolongar a conclusão da conexão e consumir os recursos do servidor. Como

o servidor fica esperando pela conclusão da requisição, os demais usuários do ambiente enfrentarão lentidão, já que o ataque deixará as conexões intermitentes.

Medidas Dentre as medidas básicas, recomenda-se rejeitar métodos HTTP não utilizados, limitar o tamanho do corpo das mensagens, dos POSTs, ajustar um timeout absoluto para as requisições, não segurar conexões rejeitadas por longos períodos e definir um fluxo mínimo de entrada de dados, matando sessões excessivamente lentas. Para exemplificar, campos de formulários que suportam entradas de até 256 caracteres não devem aceitar requisições maiores que 1 KB.

Configurações recomendadas para servidores web baseados em **Apache**:

- Usar somente as diretrizes `<Limit>` e `<LimitExcept>` para rejeitar requisições com métodos não suportados não resolve o problema, já que o Apache aguarda a requisição completar antes de processar as diretrizes. Esses parâmetros devem ser usados juntamente com os parâmetros *LimitRequestFields, LimitRequestFieldSize, LimitRequestBody, LimitRequestLine e LimitXMLRequestBody*. A configuração padrão do Apache aceita cabeçalhos com 8190 bytes, corpos com tamanho indefinido e até 100 cabeçalhos por requisição, valores excessivos para servidores convencionais.

- Reduzir os valores de *TimeOut* e *KeepAliveTimeOut*. O valor padrão de 300 é consideravelmente alto.

- O valor padrão de *ListenBackLog* (511) pode ser incrementado, ajudando nos casos em que o servidor não consegue aceitar conexões rápido suficiente.

- Ajustar o parâmetro MaxRequestWorkers para que o servidor trate o número máximo de conexões simultâneas suportadas.

- Ajustar a diretriz AcceptFilter, que permite o SO otimizar as sockets que escuta conforme o tipo de protocolo.

- Vários módulos de Apache também estão disponíveis para mitigar ataques de Slow HTTP, como o *mod_reqtimeout*, que ajusta a entrada *RequestReadTimeout* para controlar conexões lentas, configurando um tempo limite e uma taxa mínima de dados para as requisições.

Algumas configurações recomendadas para servidores web baseados em **Nginx**:

- Limitar os métodos aceitos ajustando a variável *$request_method*.

- Ajustar valores baixos para *client_max_body_size, client_body_buffer_size, client_header_buffer_* *large_client_header_buffers*.

- Atribuir valores pequenos de client_body_timeout e client_header_timeout.

- Editar *HttpLimitReqModule* e *HttpLimitZoneModule* para limitar o número de requisições e o número de conexões simultâneas para uma dada sessão e/ou endereços iguais.

- Configurar os parâmetros *worker_processes* e *worker_connections* baseados no número de CPUs / cores, conteúdo e volume de tráfego. A fórmula base adotada é *max_clients = worker_processes * worker_connections*.

Upload Irrestrito de Arquivos

A possibilidade de upload de arquivos arbitrários pode representar sérios riscos à segurança da aplicação. A funcionalidade de subida de arquivos permite a criação de componentes auxiliares para ataques.

As consequências de upload irrestrito de arquivos pode variar, incluindo o controle total do sistema sobrecarregamento do sistema de arquivos, encaminhamento de ataques para sistemas de backend ou até mesmo a pixação/modificação de conteúdo do site. A complexidade e a extensão do ataque depende de como a aplicação lida com os arquivos, seus tipos/formatos e onde armazena os mesmos.

Existem duas classes diferentes de problemas. A primeira é relacionado aos metadados dos arquivos como o nome, extensão e caminho. Caso não exista um devido controle e validação destes elementos, pode-se sobrescrever os componentes existentes ou armazená-los em locais arbitrários.

A segunda classe de problemas está relacionada ao conteúdo dos arquivos. Esta gama de problemas depende diretamente do uso e formato de armazenamento dos arquivos por parte da aplicação. Para proteger contra estes tipos de ataques, deve-se analisar o conteúdo de todos arquivos recebidos pela aplicação, atentando para as operações de processamento, armazenamento e interpretação do conteúdo envolvido.

Impacto As consequências do Upload Irrestrito de Arquivos podem variar, incluindo aquisição completa do sistema, o sobrecarregamento do sistema de arquivos, encaminhando ataques a sistemas de back-end, e defacement. O impacto também dependerá do que o aplicativo faz com o arquivo enviado, incluindo onde ele é armazenado.

Medidas Validação de tipo de arquivo: para determinadas situações, aplicações Web usam implementações internas para validar o formato de tipos de arquivos processados. Supondo que a aplicação realize alguma operação com imagens (redimensionamento, por exemplo), validar e reconhecer o arquivo enviado auxilia para bloquear um conteúdo não permitido.

Alguns reconhecedores lêem os primeiros caracteres do arquivo (cabeçalho) para realizar essa validação. Neste caso, um atacante pode inserir um conteúdo arbitrário malicioso após esse cabeçalho aceito.

Existem alguns locais específicos da estrutura dos arquivos destinados a seções de comentários que não afetam a estrutura básica do arquivo. Esses campos também podem ser abusados para inserir conteúdo malicioso. Deve-se atentar também para as bibliotecas que

realizam as operações auxiliares nos arquivos (como a *libmagick*[5]), que podem se deparar com um código malicioso e se comportar de maneira não esperada.

Outras medidas são:

- Nunca deve-se aceitar extensões de tipos de arquivo não inclusos em um filtro prévio (white-list).
- Todos os caracteres de controle Unicode deve ser removidos dos nomes de arquivo e suas extensões, sem exceção. Caracteres especiais como “;”, “:”, “>”, “<”, “/” ,”\”, além de repetidos “.”, “*”, “%” e “$” devem ser descartados. De maneira geral, recomenda-se a restrição no nome para caracteres alfanuméricos, não permitindo extensões vazias, como na express regular [a-zA-Z0-9]1,200\.[a-zA-Z0-9]1,10.
- Deve-se limitar o tamanho de nome do arquivo. O tamanho máximo permitido do nome do arquivo com sua extensão não deve exceder 255 caracteres (desconsiderando o diretório) em uma partição NTFS, por exemplo. Caso a aplicação não saiba lidar com nomes maiores que o permitido, ela pode causar uma condição de negação de serviço.
- Recomenda-se usar um algoritmo/lógica interna para determinar os nomes de arquivos a serem armazenados (um hash seguido da data/timestamp, por exemplo).
- Os diretórios de Upload nunca devem ter a flag de execução permitida.
- Os tamanhos de arquivos devem ser limitados a um valor razoável para evitar ataques de negação de serviço (no uso de disco, banda, processamento).
- Manter um controle de requisições, através de métodos anti-XSRF (proteção contra *Cross Site Request Forgery*).
- Deve-se validar e não sobrescrever imagens com um mesmo hash.
- Métodos POST devem ser preferíveis a PUT ou GET.
- As atividades devem ser registradas, mantendo-se um log das operações internamente.

Para maiores informações: `https://www.owasp.org/index.php/Unrestricted_File_Upload`

Ausência de Controle de Requisições

A ausência de controle de requisições ocorre, por exemplo, quando o usuário gera acessos no banco de dados sem nenhuma limitação por parte da aplicação, possibilitando assim, que um usuário envie múltiplas requisições com combinações alternadas. O usuário pode abusar enviando múltiplas perguntas e imagens para o banco de dados.

[5]Mais informações em: `http://www.imagemagick.org/`

Impacto O sistema não realiza nenhuma validação adicional durante a submissão de dados do usuário. Um usuário malicioso pode explorar o ambiente de várias maneiras:

- É possível capturar o conteúdo do POST durante uma submissão e repetir as entradas por um número indeterminado de vezes até sobrecarregar o ambiente, derrubando o banco de dados e causando negação de serviço. Outra possibilidade seria abusar da interface administrativa de moderação enviando um número indefinido de perguntas, dificultando o trabalho dos moderadores.

- Como o conteúdo dos campos não é verificado, é possível povoar a página com palavras arbitrárias, propagandas, URL's e scripts maliciosos, por exemplo. Estes termos de propaganda podem ser indexados pelo Google e usados para *comment spam*. É importante observar que, caso o banco de dados seja compartilhado com outros ambientes, o abuso das requisições pode causar uma condição de negação de serviço em toda plataforma que faz uso das bases.

Medidas É altamente recomendável que exista um controle do número total de requisições partindo de um mesmo usuário. É possível criar controles internos na aplicação para restringir e limitar requisições repetidas partindo de uma mesma combinação de endereço IP e *User Agent*.

Uma solução comum adotada pelos desenvolvedores contra CSRF e que pode ser aproveitada nesse caso é o *Synchronizer Token Pattern*, que gera um token de 'desafio" aleatório associado com a sessão atual do usuário. Esses *tokens* são inseridos nos formulários HTML, links associados com operações sensiveis e incluídas nas solicitações HTTP quando enviadas para o aplicativo web.

Quando o usuário gera uma solicitação por meio de um formulário, um campo "input" do tipo 'hidden" deverá conter o *token* de proteção CSRF. Toda vez que uma transação sensível for executada, o aplicativo irá comparar o token enviado com o valor armazenado em sessão. Caso positivo, a transação será realizada e um novo *token* gerado de maneira aleatório. Em caso negativo, o aplicativo deverá recusar a execução da transação.

Uma requisição externa não consegue tomar conhecimento deste *token* identificador sem interagir com a página. Nesse caso, o ataque precisaria ser mais elaborado para obter o *token* desafio para então submeter a requisição.

Outra possibilidade para esse caso seria usar a sessão associada dos usuários que entram e autenticam. A partir daí, pode-se limitar as requisições conforme a sessão única, que expira após um determinado tempo.

O uso de *CAPTCHA's*[6] também auxilia limitando o uso de scripts e comandos automatizados para submissão de respostas. Para casos em que a interação do usuário deve ser rápida e simples, é possível usar uma abordagem que exige a resolução do CAPTCHA

[6]CAPTCHA é um acrônimo da expressão "Completely Automated Public Turing test to tell Computers and Humans Apart"(teste de Turing público completamente automatizado para diferenciação entre computadores e humanos): um teste de desafio cognitivo, utilizado como ferramenta anti-spam. Mais informações em: `http://www.captcha.net/`

mediante situações de estresse, quando partem várias requisições de um mesmo endereço IP por exemplo.

Falha na Restrição de Acesso à URLs

Uma Falha na Restrição de Acesso à URLs ocorre quando existe uma falha no controle de acesso da aplicação e um usuário sem privilégios acessa recursos administrativos da aplicação.

Impacto Tais falhas podem permitir que algumas ou até mesmo todas as contas possam ser comprometidas. Uma vez bem sucedido, o atacante pode fazer qualquer coisa com os privilégios do usuário comprometido. Contas privilegiadas são frequentemente alvo desses ataques.

Medidas Impedir o acesso não autorizado URL requer a devida autenticação e autorização para cada página. Frequentemente, esta proteção é proporcionada por um ou mais componentes externos para o código de aplicação. Independentemente do mecanismo, todos os seguintes são recomendados:

- As políticas de autenticação e autorização devem ser baseada no papel de cada usuário, minimizando assim o esforço necessário para manter estas políticas.
- As políticas devem ser altamente configuráveis, a fim de minimizar quaisquer aspectos difíceis da política.
- O mecanismo de aplicação deve negar todo o acesso por padrão, exigindo subsídios explícitos para os usuários e papéis específicos para acesso a cada página.
- Se a página está envolvido em um fluxo de trabalho, verifique se as condições estão no estado adequado para permitir o acesso.

Capítulo 7

OWASP ZAP

Conforme mencionamos na página 56 o OWASP ZAP é um proxy de interceptação com diversos recursos para auditoria de aplicações web.

A principal filosofia envolta no desenvolvimento da ferramenta é que "Você não pode construir aplicações seguras se não conhece como elas são atacadas". Assim, a solução encontrada foi desenvolver uma ferramenta de fácil usabilidade, onde os desenvolvedores poderiam testar suas próprias aplicações.

7.1 Histórico

O OWASP ZAP foi lançado em Setembro de 2010[1] com a premissa de ser uma ferramenta com ampla facilidade de uso e boa documentação.

A ferramenta é open-source, com suporte a internacionalização e com diversos idiomas em sua base, inclusive com suporte ao português.

Finalmente, a ferramenta é multi-plataforma, escrita em Java. Possui versões lançadas com regularidade e uma versão lançada toda semana com as últimas modificações. Também possui uma ampla variedade de extensões[2] que extendem suas verificações, assim ampliando sua cobertura de testes.

7.2 Preparação

Para utilizar o OWASP ZAP de forma efetiva, é necessário configurar o navegador de maneira a não interferir no tráfego. No exemplo de configuração, vamos abordar o navegador Mozilla Firefox[3]

[1] A apresentação completa do projeto está disponível em: `http://www.owasp.org/images/c/c8/Conference_Style_slides_for_ZAP.ppt`

[2] Informações sobre extensões podem ser encontradas em: `https://code.google.com/p/zap-extensions/`

[3] Disponível em: `http://www.mozilla.org/pt-BR/firefox/new/`

É válido lembrar que esse navegador deve ser atualizado manualmente e, sempre quando possível, não ser usado para uso cotidiano. O ideal é que o navegador seja usado somente para uso com o OWASP ZAP.

7.2.1 Desative a atualização das extensões

Durante a navegação normal e ao iniciar o navegador, seu sistema de extensões busca por atualizações das mesmas de modo a proporcionar uma melhor experiência de uso. Para uso com o OWASP ZAP, esse tráfego de rede insere elementos que podem poluir a visibilidade do funcionamento da aplicação web.

Para desativar esse recurso, vá em `about:config`, use o recurso localizar para encontrar os itens *app.update.enabled* e *extensions.update.enabled*. Em seguida coloque seus valores para *false*, conforme demonstrado na imagem 7.1.

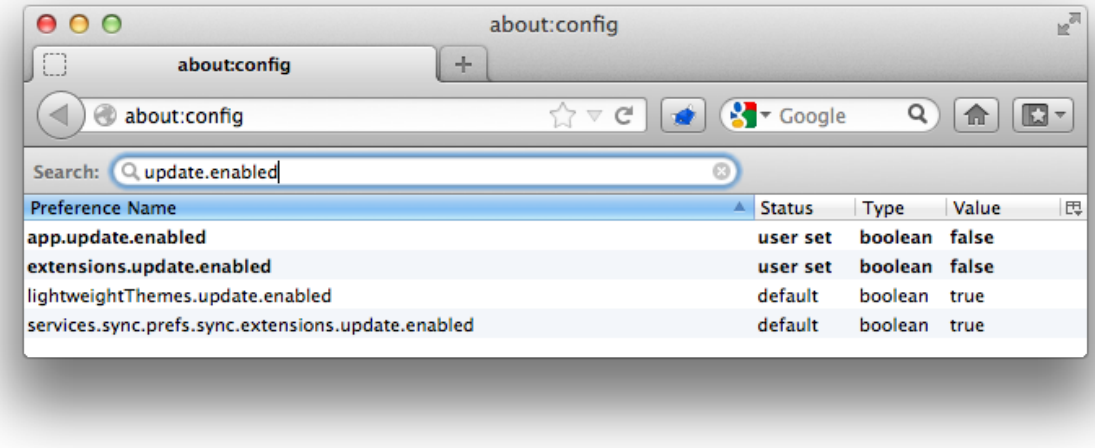


Figura 7.1: Parâmetros de atualização das extensões em negrito

7.2.2 Desative o SafeBrowsing

Durante a navegação cotidiana, o navegador verifica se os endereços requisitados pertencem a uma lista de endereços conhecidamente perigosos. Para uso com o OWASP ZAP, esse tráfego de rede insere elementos que podem poluir a visibilidade do funcionamento da aplicação web.

Para desativar esse recurso, vá em `about:config`, use o recurso localizar para encontrar os itens *browser.safebrowsing.enabled*, *browser.safebrowsing.malware.enabled*, *services.sync.prefs.sync.browser.* e *services.sync.prefs.sync.browser.safebrowsing.malware.enabled*. Em seguida coloque seus valores para *false*, conforme demonstrado na imagem 7.2.

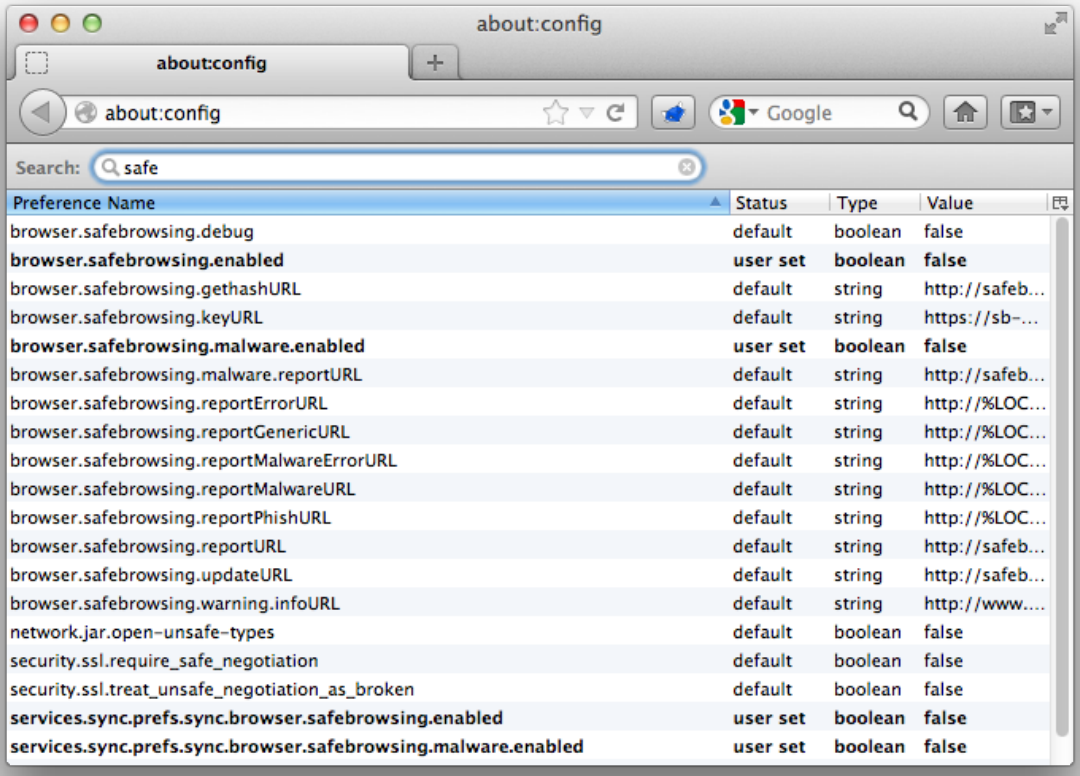


Figura 7.2: Parâmetros do SafeBrowsing em negrito

7.2.3 Desative o HTTP Pipelining

O *HTTP Pipelining* é uma técnica que permite enviar múltiplas requisições através de uma conexão TCP estabelecida com o servidor. Isso ocorre de maneira assíncrona, ou seja, o cliente não precisa esperar a resposta da requisição feita para enviar a próxima requisição[4].

O uso desse recurso em conjunto com o OWASP ZAP pode impactar na usabilidade da ferramenta. Para desativar esse recurso, vá em `about:config`, use o recurso localizar para encontrar os itens *network.http.pipelining* e *network.http.pipelining.ssl*. Em seguida coloque seus valores para *false*, conforme demonstrado na imagem 7.2.

[4]Mais informações disponíveis em: `http://www.w3.org/Protocols/rfc2616/rfc2616-sec8.html#sec8.1.2.2`

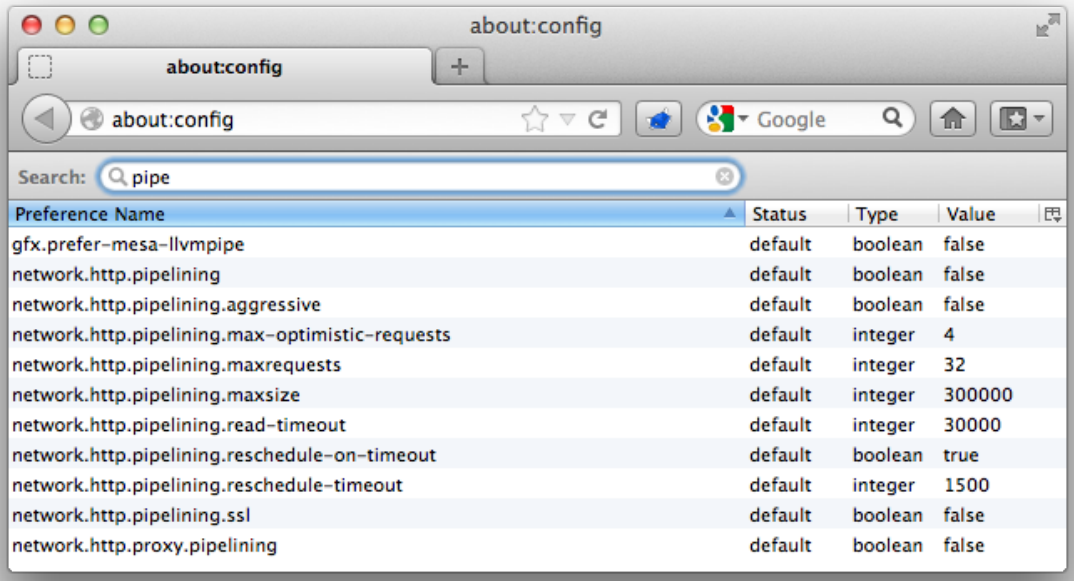


Figura 7.3: Parâmetros do Pipelining em negrito

7.2.4 Desative o Prefetching

O *Prefetching* é um recurso do navegador que usa o tempo ocioso do navegador para realizar o download de páginas e documentos que o usuário possa vir a clicar. Algumas páginas fornecem configurações especificando quais conteúdos podem ser descarregados através desse recurso de modo que seu funcionamento pode ser involuntario, desde que o navegador esteja ocioso[5].

O uso desse recurso em conjunto com o OWASP ZAP pode impactar na poluição dos dados mostrados na ferramenta. Para desativar esse recurso, vá em `about:config`, use o recurso localizar para encontrar os itens *network.dns.disablePrefetch* e *network.prefetch-next*. Em seguida coloque seus valores para *true* e *false*, respectivamente, conforme demonstrado na imagem 7.2.

7.2.5 Mude a página inicial

Ao abrir o navegador ele automaticamente carrega a página inicial que foi previamente configurada. Frequentemente, é possível que ao executarmos o navegador, o OWASP ZAP já esteja em execução, de modo que os dados carregados pela página poluam os dados coletados anteriormente com o OWASP ZAP.

[5]Mais informações em: `https://developer.mozilla.org/en-US/docs/Link_prefetching_FAQ`

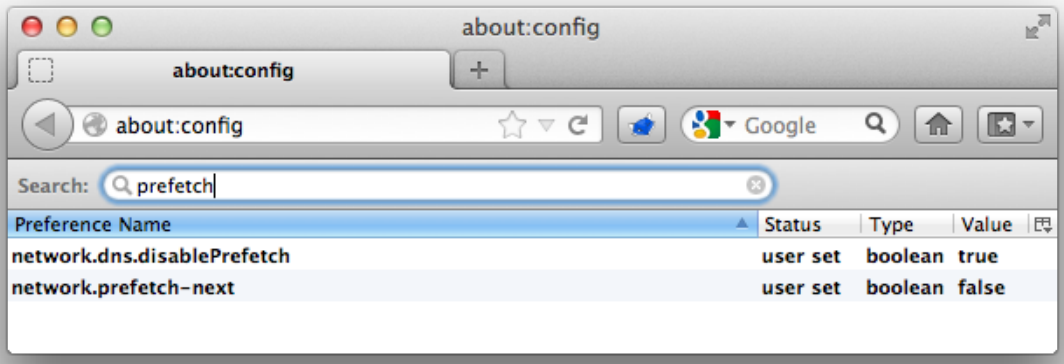


Figura 7.4: Parâmetros do Pre-fetching em negrito

Para um melhor uso desse recurso é interessante que o navegador não carregue páginas ao iniciar, para isso vá em *Preferências*, na aba *Geral* e configure a função inicial do navegador para *Abrir página em branco*, conforme demonstrado na imagem 7.2.

7.2.6 Desative as sugestões de página

Para prover uma melhor experiência de uso, assim que o usuário começa a digitar algum conteúdo, o navegador com base em suas preferências sugere frases ou domínios que possam ser de interesse do usuário.

Esses dados, obtidos através da coleta de informações de navegação ou de informações de terceiros, como o Google, por exemplo, podem interferir nos dados que são enviados para a aplicação web que estará sendo testada. Além disso, caso esses dados sejam provenientes de terceiros, o tráfego envolvido ainda poderá poluir os registros da aplicação.

Logo o uso desse recurso em conjunto com o OWASP ZAP pode impactar na poluição dos dados mostrados na ferramenta assim como nos dados enviados para a aplicações web envolvida. Para desativar esse recurso, vá em *Preferências*, na aba *Privacidade* e configure em *Campo de endereço* a opção *Nada*, conforme demonstrado na imagem 7.6.

7.2.7 Desabilite os updates do navegador

Periodicamente, o navegador entra em contato com seu fabricante de modo a obter informações sobre atualizações e novas versões. A finalidade desse recurso é prover ao usuário a melhor experiência possível em navegação web, assim permitindo que ele sempre desfrute dos últimos recursos disponíveis, tanto para a navegação quanto em termos de segurança.

O uso desse recurso em conjunto com o OWASP ZAP pode impactar na poluição dos

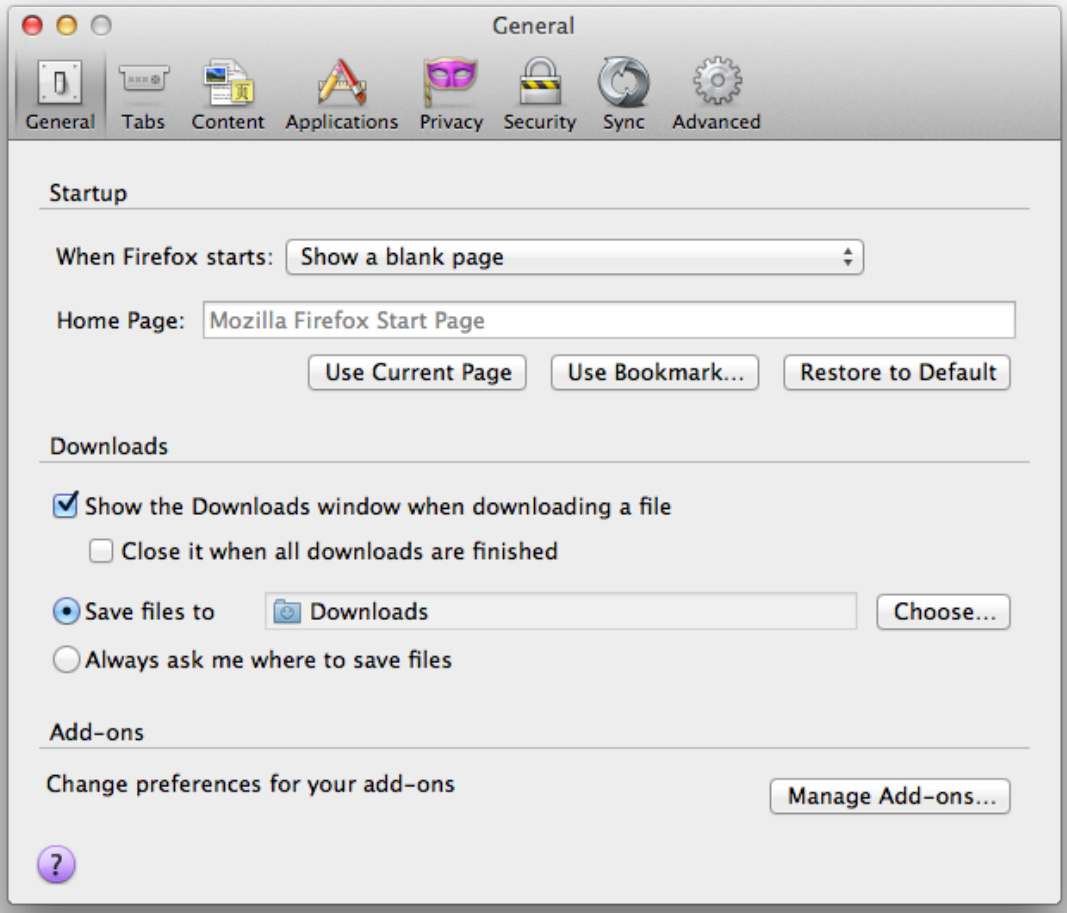


Figura 7.5: Configuração Geral do Firefox

dados mostrados na ferramenta, pois periodicamente o navegador poluirá os dados com requisições buscando informações sobre atualizações. Para desativar esse recurso, vá em *Preferências*, na aba *Avançado* e configure em *Atualizações* a opção *Nunca verificar*, conforme demonstrado na imagem 7.7.

7.2.8 Habilitando a opção de proxy

Ao utilizar o Firefox com o OWASP ZAP, é necessário configurar o navegador para encaminhar suas requisições para o OWASP ZAP de maneira que ele seja capaz de interceptar o tráfego HTTP.

Para utilizar esse recurso, vá em *Preferências*, na aba *Avançado* e configure em *Rede* a

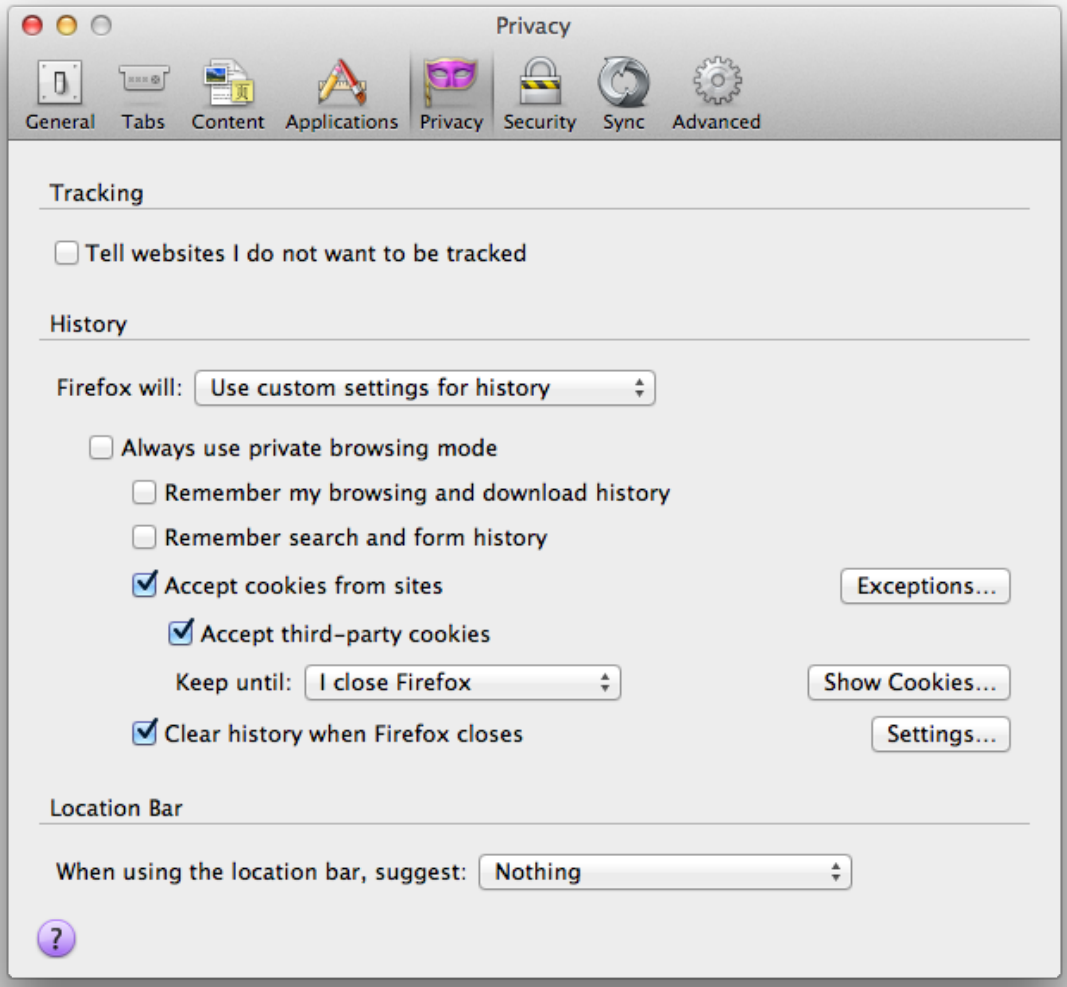


Figura 7.6: Configuração de Privacidade do Firefox

opção *Conexão* para *Configuração Manual de Proxy* com os campos *Endereço* e *Porta* com os valores *localhost* e *8080*[6], respectivamente.

[6]Os valores podem alterar conforme a configuração do sistema. Consulte a seção 7.3.1 para saber qual a porta que o OWASP ZAP estará usando em seu sistema

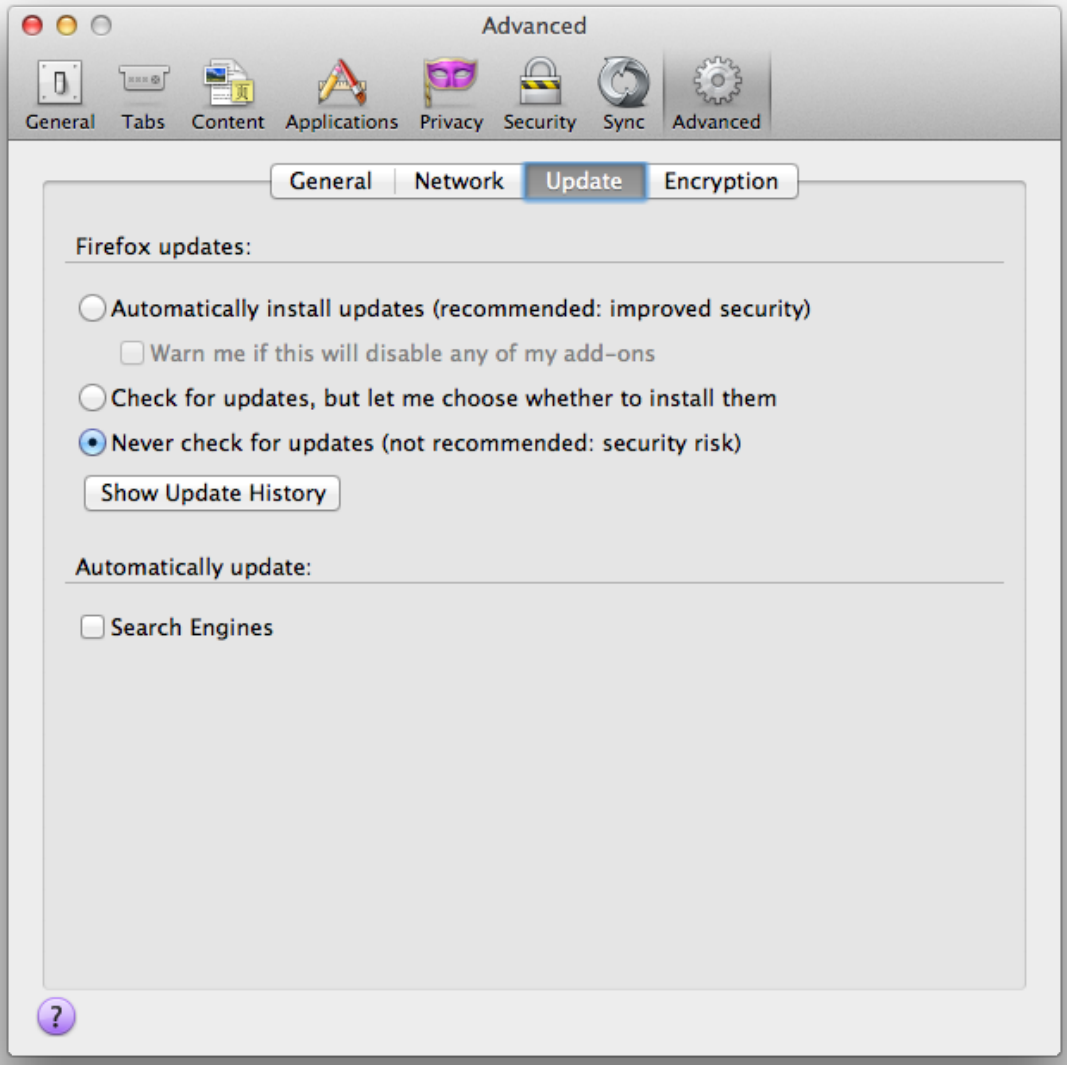


Figura 7.7: Configuração Avançada do Firefox

7.3 Usando a ferramenta

O OWASP ZAP é uma ferramenta rica em recursos úteis para auditoria em aplicações web. Veremos a seguir como utilizar alguns de seus recursos.

7.3.1 Configuração Inicial

Após iniciar a ferramenta, o primeiro passo é realizar sua configuração inicial. Para isso, vá no menu *Ferramentas* e selecione *Opções*[7]. Confira na imagem 7.8 a janela de opções.

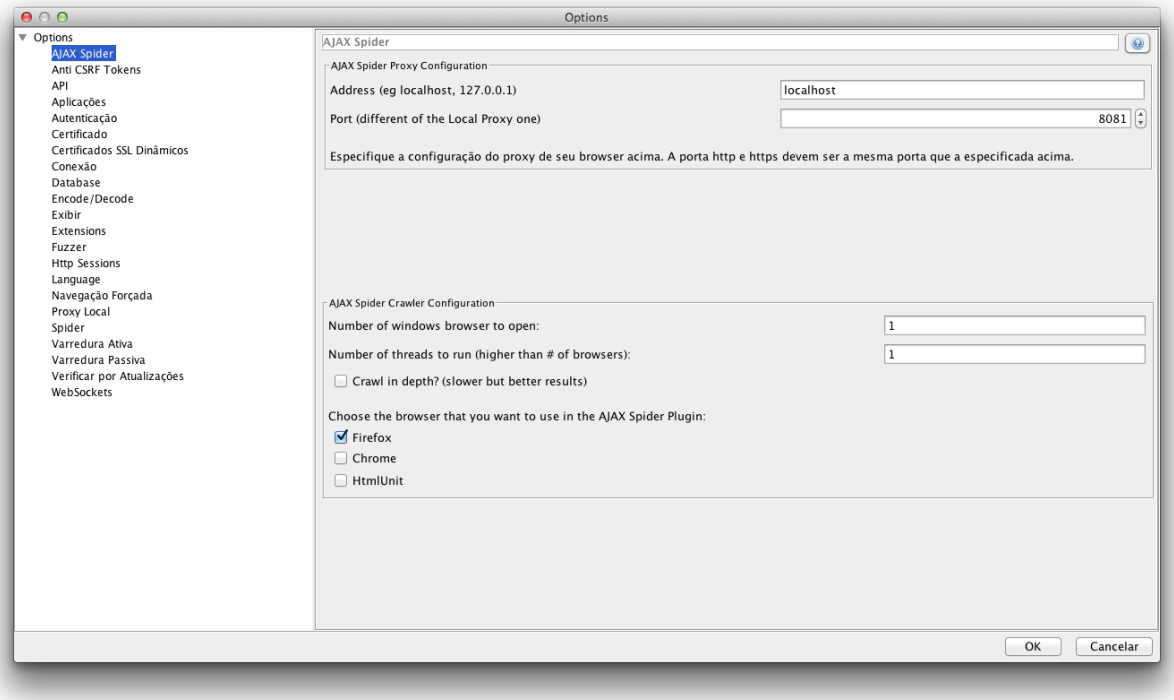


Figura 7.8: Janela de opções do OWASP ZAP

A primeira configuração a ser realizada é a verificação da porta de funcionamento da ferramenta. Para isso, selecionaremos a opção *Proxy local*, conforme ilustrado na imangem 7.9.

Logo após, devemos configurar a geração de certificados SSL na ferramenta. Essa funcionalidade permite ao OWASP ZAP interceptar o tráfego criptografado através da geração de certificados dinâmicos. Para tal, devemos importar o certificado raiz gerado pela ferramenta no nosso sistema e configurar tal certificado como *confiável*.

Na mesma janela, selecionaremos a opção *Certificados SSL Dinâmicos* conforme ilustra a figura 7.10.

Caso seja de interesse, o idioma da ferramenta pode ser alterado na opção *Language*, conforme ilustrado na imagem 7.11.

Outra configuração interessante é a de habilitar a transformação de campos do tipo *hidden* em campos do tipo *text*, o que permite sua visualização e alteração durante a navegação da

[7]É possível realizar o mesmo procedimento selecionando na interface da ferramenta o icone com a figura de uma engrenagem.

Figura 7.9: Opções de Proxy do OWASP ZAP

aplicação. Para habilitar esse recurso, selecione o botão na barra de ferramentas do OWASP ZAP com a imagem de uma lâmpada.

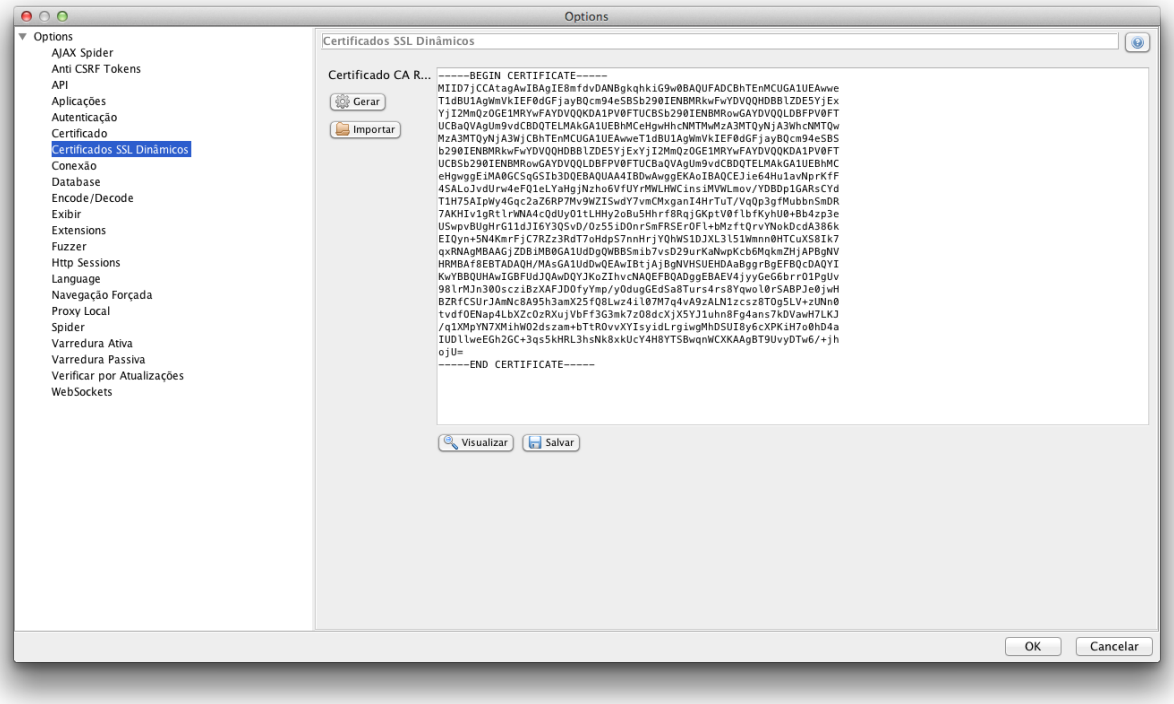


Figura 7.10: Opções de Certificados SSL Dinâmicos do OWASP ZAP

7.3.2 Funcionalidades Basicas

Agora com a ferramenta configurada, veremos um pouco de sua funcionalidade como *proxy de interceptação*. Para isso, basta navegar um pouco com o Firefox em algum site de sua escolha.

Ao voltar para a janela do OWASP ZAP veremos que novas informações estão disponíveis. Todo o tráfego HTTP utilizado na navegação estará disponível para inspeção. Por padrão, o OWASP ZAP não mostra as imagens carregadas em sua interface de maneira a reduzir a quantidade de informações que devem ser analizadas. Caso a imagem receba algum parâmetro, ela será mostrada.

Veja na imagem 7.12 como ficou a janela do OWASP ZAP após navegar por alguns minutos.

Agora com os dados da navegação, podemos verificar outras funcionalidades da ferramenta através da navegação entre suas abas. No canto superior direito, temos a opção de ver a requisição e a resposta realizada para cada recurso que foi acessado, assim como os dados enviados e recebidos no corpo da requisição e resposta.

No canto superior esquerdo podemos acompanhar de forma estruturada os recursos acessados durante a navegação.

No parte inferior, na aba *Histórico* podemos acompanhar as requisições realizadas por ordem de execução. Ainda na parte inferior, ao mudarmos para a aba:

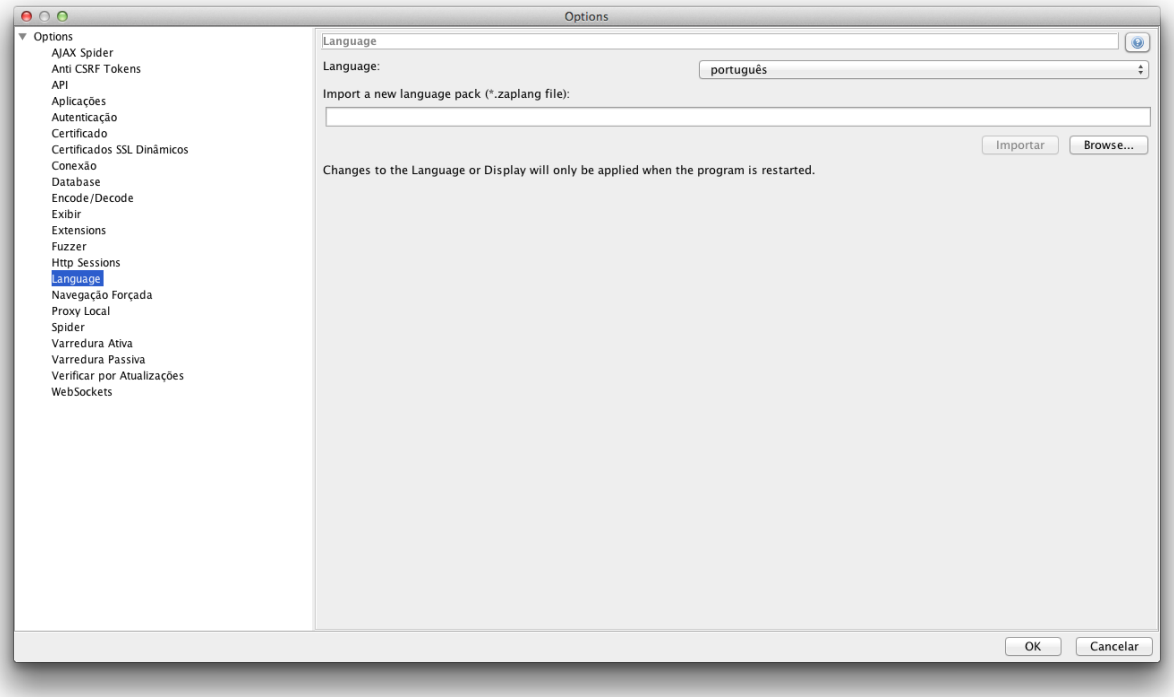


Figura 7.11: Opções de Idioma do OWASP ZAP

Localizar podemos procurar por textos em todo o conteúdo presente, ou fazer buscas seletivas por textos presentes na URL, Requisição, Resposta, Cabeçalhos ou nos resultados do Fuzz.

Alertas podemos acompanhar os alertas automáticos emitidos pela ferramenta. Os alertas ficam estruturados de acordo com seu risco, representado pela cor[8] da bandeira que acompanha sua descrição na coluna da esquerda.

Maiores detalhes podem ser obtidos selecionando o item e visualizando as informações na coluna da direita.

Ao clicar duas vezes no item da coluna da esquerda, é possível alterar seus parâmetros e opções, assim como os demais campos.

Spider podemos realizar uma varredura recursiva na aplicação de forma a mapearmos praticamente todo o seu conteúdo.

O *spider* funciona através do uso das URLs que aparecem nas páginas durante a navegação. Elas são utlizadas para pegar o conteúdo de outras páginas e com isso aumentar a base de URLs a serem visitadas. Para que o *spider* funcione de maneira eficaz, é necessário uma navegação diversificada na aplicação.

[8]Azul: Informação, Amarelo: Baixo, Laranja: Médio e Vermelho: Alto.

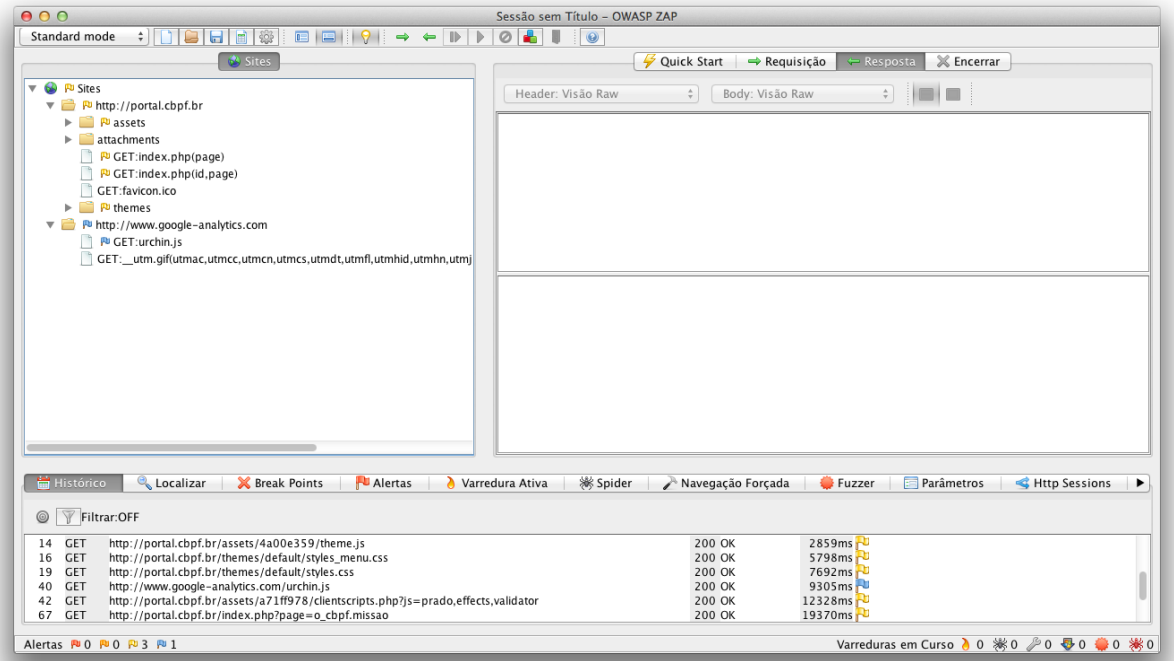


Figura 7.12: Janela do OWASP ZAP após navegar por alguns minutos

Também é possível usar essa funcionalidade através do quadro superior direito da aplicação, selecionando o item e abrindo o menu de opções, selecionar *Ataque* e logo em seguida *Usar o sipder no site*.

Navegação Forçada podemos selecionar listas para descobrirmos diretorios e recursos até então não mapeados anteriomente. Esses recursos podem ser backups, versoes antigas de arquivos renomeados, diretórios desprotegidos entre outros.

Também é possível usar essa funcionalidade através do quadro superior direito da aplicação, selecionando o item e abrindo o menu de opções, selecionar *Ataque* e logo em seguida *Site com navegação forçada*.

Por padrão são fornecidos com o OWASP ZAP algumas listas de diretórios que podem ser usadas, mas a confecção de uma lista personalizada é o ideal.

Parâmetros podemos acompanhar os parâmetros identificados pela ferramenta durante a navegação e o *spider*. Esses itens são um ponto chave pois eles identificam os pontos de entrada de informação da aplicação.

7.3.3 Reenvio e Fuzzing

O OWASP ZAP possui dois mecanismos interessantes para a alteração dos dados enviados para a aplicação web. O reenvio é feito de foma individual. O usuário deve selecionar a requisição que gostaria de alterar e então fazer o reenvio dela com parâmetros escolhidos por ele.

Confira na imagem 7.13 a utlilização do recurso, que foi feita após a seleção da requisição (via histórico) e a seleção da opeção de *Reenviar*.

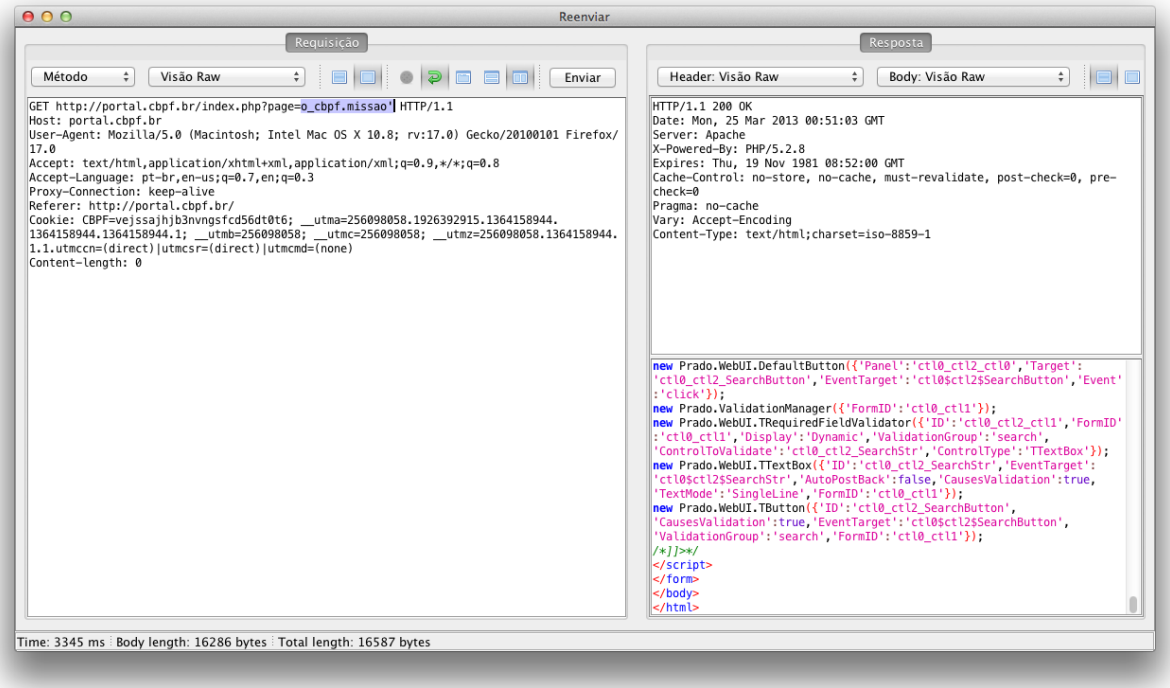


Figura 7.13: Uso da funcionalidade de reenvio no OWASP ZAP

Normalmente, desejamos fazer o reenvio diversas vezes de maneira a tentar cobrir o máximo de possibilidades possíveis. O OWASP ZAP implementa esse recurso através do uso de listas que são confeccionadas com os valores que se deseja substituir no reenvio. Para fazer uso desse recurso, selecione a requisição que deseja alterar os dados enviados. Em seguida, na aba *Requisições* selecione o pedaço do texto que gostaria de alterar. Clique sobre o texto e selecione a opção *Fuzz...*. Acompanhe na imagem 7.14 a seleção de listas.

7.3.4 Decode e Encode

Frequentemente nos deparamos com informações codificadas ao interagirmos com aplicações web. Na grande maioria essas informações estão codificadas em *base64*[9]. O OWASP

[9]Mais informações em: `http://tools.ietf.org/html/rfc4648`

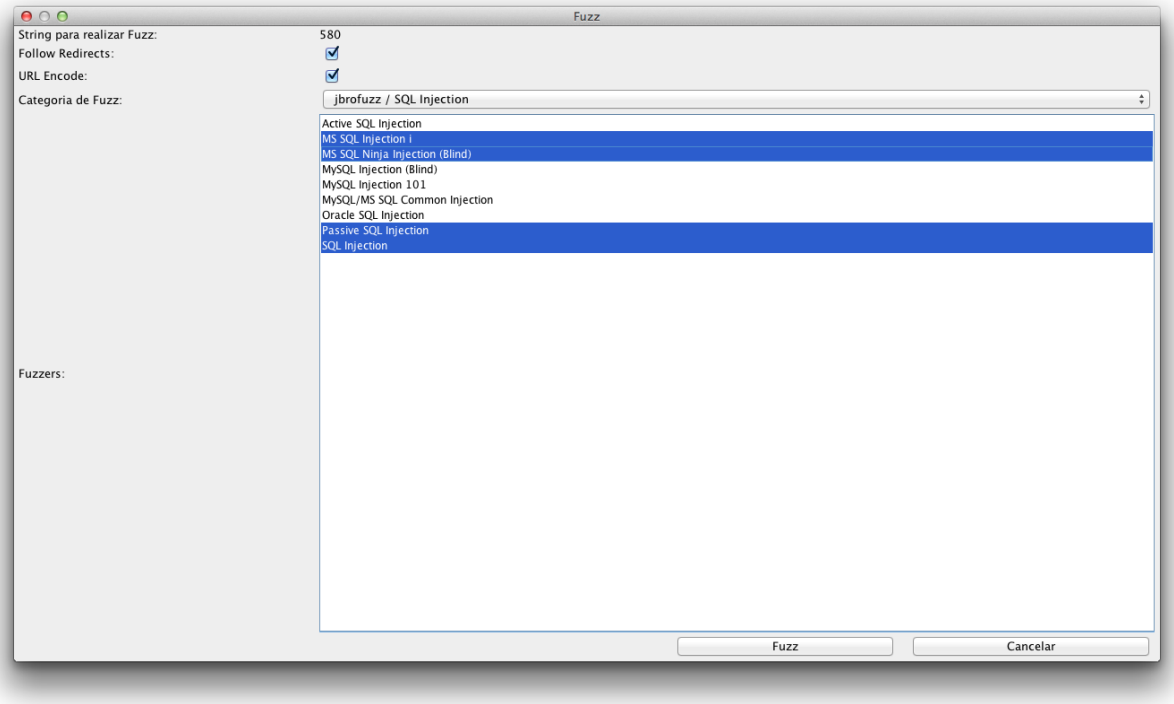


Figura 7.14: Uso da funcionalidade de *Fuzz* no OWASP ZAP

ZAP fornece uma forma ágil de decodificar essas informações. Para utilizar tal recurso, selecione a informção que deseja decodificar, ou codificar e com o menu auxiliar selecione *Codificar/Decodificar/Hash*.

Acompanhe na imagem 7.15 o uso dessa funcionalidade no OWASP ZAP.

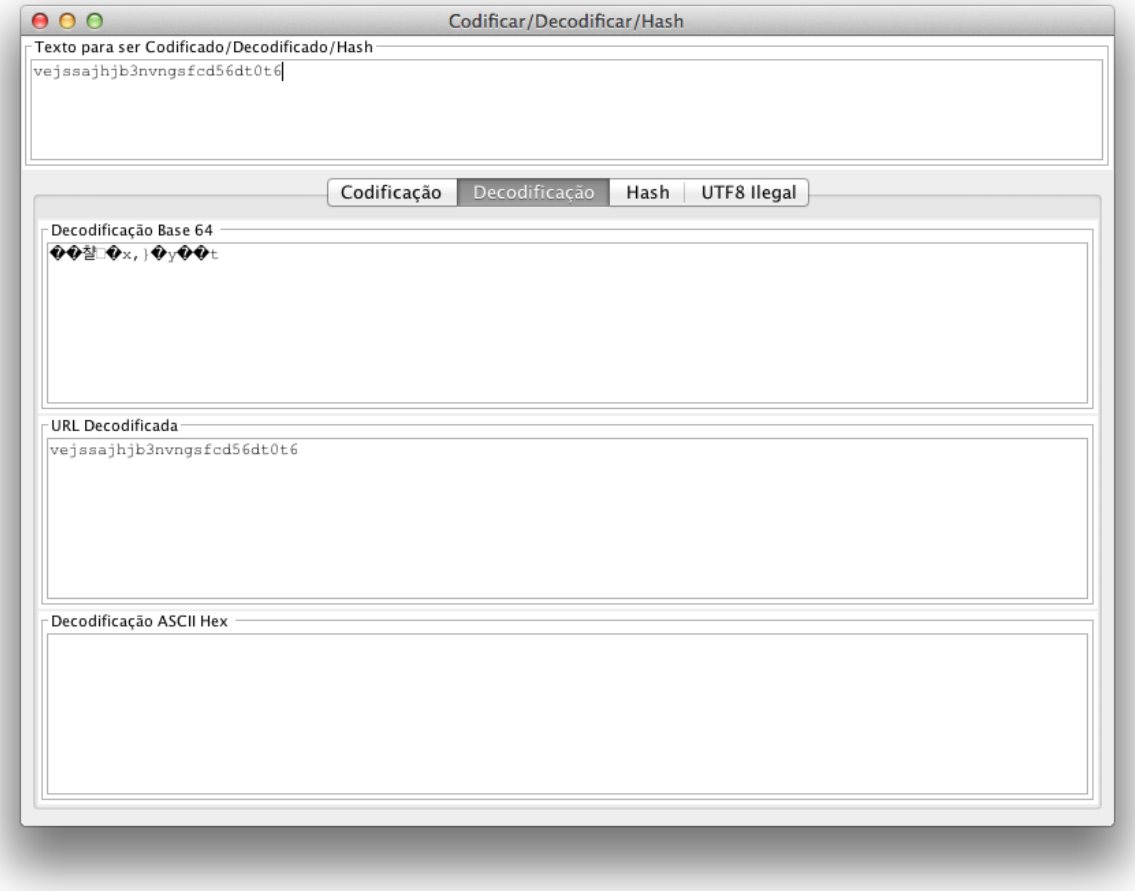


Figura 7.15: Uso da funcionalidade de *Codificar/Decodificar/Hash* no OWASP ZAP

Capítulo 8

Considerações finais

Uma auditoria fornece uma visão profunda sobre a segurança dos ativos contemplados, fundamentada tanto por informações obtidas durante a mesma ou fornecidas pelo próprio solicitante. Por sua natureza, é uma atividade que exige, do lado do solicitante, confiança para revelar informações sensíveis e permitir a realização da auditoria e, do lado do auditor, ética para lidar com informações fornecidas ou descobertas com responsabilidade.

Outra questão importante diz respeito a transparência de todo o processo da auditoria junto ao solicitante. Não se pode esperar que o solicitante vá confiar cegamente no auditor e esperar pacientemente a conclusão da auditoria para receber resultados. Assim, o solicitante deve ser informado periodicamente sobre o andamento da auditoria incluindo informações como quais procedimentos já foram realizados e quais serão executados até próximo reporte, sem nunca fugir da metodologia previamente discutida com o mesmo.

8.1 Ética

A ética é muito importante para a realização de auditorias principalmente pelo volume de informações sensíveis tratadas pelo auditor e deve ser levada em conta durante todo o processo de auditoria. A divulgação, mesmo que parcial ou acidental, destas informações pode provocar incidentes de segurança sérios nos ativos contemplados pela auditoria. O propósito da revelação destas informações é especificamente avaliar a segurança dos ativos do contratante, então o uso destas informações para qualquer outro propósito vai contra os interesses do mesmo e, mesmo na ausência do acordo, a divulgação destes dados configura um conflito ético.

Outro conflito ético acontece quando são realizados auditorias não requisitadas. Esta prática consiste da realização da auditoria sem o prévio conhecimento ou autorização do alvo da mesma. Sob uma ótica puramente técnica, as auditorias não-requisitada e caixa preta se dão exatamente da mesma forma, diferindo somente na apresentação do resultado.

Enquanto os resultados de auditorias caixa preta consistem principalmente da demonstração de evidências dos problemas de segurança encontrados e medidas de mitigação para os mesmo, os resultados de auditorias não-requisitadas geralmente consistem somente das

evidências dos problemas de segurança encontrados, de forma que a entrega do restante das informações fica condicionada uma ação determinada pelo autor da auditoria. Esta ação varia muito dependendo dos interesses de quem realizou a auditoria, indo de pagamento do serviço sob pena de divulgação das informações coletadas em meios de comunicação públicos até contratação do mesmo na empresa sob o pretexto de resolver os problemas de segurança encontrados.

As práticas supracitadas são consideradas anti-éticas e, em alguns casos, até criminosas, embora algumas nem sempre tenham sido classificadas assim. No mercado da segurança da informação, a reputação de um profissional ou empresa é algo valioso e fazer uso de qualquer uma destas práticas deteriora esta reputação, em alguns casos irremediavelmente. É importante que o analista, ao se propor a fazer uma auditoria, tenha em mente que a ética profissional é o que o separa dos criminosos cibernéticos e que o propósito da auditoria deve ser medir a resistência do sistema a ataque e nunca ver quanto dano se pode causar com um ataque. Desta forma, quando uma simulação de ataque é bem-sucedida, é responsabilidade do analista avaliar se deve utilizar o fruto desta para lançar outras ou se este resultado basta para cumprir o objetivo da auditoria.

> Aquele que se empenha a resolver as dificuldades resolve-as antes que elas surjam. Aquele que se ultrapassa a vencer os inimigos triunfa antes que as suas ameaças se concretizem.
>
> *Sun Tzu*

