****

**Visão**

Com a crescente demanda sobre Tecnologias, percebemos que muitas pessoas apesar de buscarem informações, não possuem fontes que queiram realmente passar o conhecimento da maneira como ela deve ser, livre e com embasamento técnico que permita ser aplicado e utilizado quando necessário, além de serem testados em sua criação, tornando esta informação útil e confiável.

**Missão**

O Laboratório foi criado com a intenção de buscar e disseminar o conhecimento de uma maneira clara e objetiva, de forma gratuita, auxiliando na evolução dos membros e da sociedade na qual estas informações são compartilhadas, buscando o crescimento de todos os envolvidos nesta criação de valores.



Caso você pense que com a leitura dos materiais da How2Security, você irá se tornar um Cracker capaz de invadir sistemas, se você espera encontrar aqui scripts infalíveis para invasão e, a partir deles, sair por aí invadindo computadores, essa não é a leitura indicada. Indicamos, sim a leitura do Código Penal (Lei 2.848/1940), principalmente a Lei Carolina Dickmann (Lei 12.737/2012), nos Artigos 154-A e 154-B.

*154-A Invadir dispositivo informático alheio, conectado ou não à rede de computadores, mediante violação indevida de mecanismo de segurança e com o fim de obter, adulterar ou destruir dados ou informações sem autorização expressa ou tácita do titular do dispositivo ou instalar vulnerabilidades para obter vantagem ilícita:*

*Pena – Detenção, de 3 meses a 1 ano, e multa*

Este material é um conjunto de informações compiladas de documentos e ferramentas do Mundo Underground testadas em ambiente de laboratório na nossa intranet. Desta forma, todo conhecimento aqui condensado é tangível, assim como as orientações das contramedidas.

Dessa forma, esperamos ter sido bem claros que, em momento algum, estamos com a pretensão de ensinar a você como se tornar um invasor. Estaremos sim, mostrando muitas das técnicas utilizadas pelos crackers e, em alguns casos, pelos scripts kiddies, para que você, como administrador de redes, seja capaz de identificá-las em tempo hábil para se defender, antes que alguém com desejos menos nobres ô faça por você.

Assim sendo, todo o conteúdo dessa literatura tem apenas o objetivo didático de informar e preparar os administradores de redes dos novos tempos. Em momento algum nos responsabilizamos pelo mau uso desse conhecimento ou por danos causados em seu equipamento ou de terceiros, assim como também não somos responsáveis pelos códigos e ferramentas aqui citados.

Sandro Melo

Adaptado por Wellington Silva aka Well

**0 – Recon Ativo – Spider Crawling em WebSites**

Várias ferramentas podem fazer o spidering automaticamente, requisitando as páginas web procurando por links para outros conteúdos de forma recursiva.

As funções dessas ferramentas é percorrer a aplicação web, submeter vários dados (de forma randômica) em formulários HTML, procurando Drop-Down, Radio-Box, Check-Box, entre outros, e ver as variáveis passadas na URL.

Algumas dessas ferramentas incluem Burp Suite, WebScarab, Zed Attack Proxy, entre outras.

A maioria dos servidores web tem um arquivo de nome ***robots.txt*** que contém uma lista de URLs que não devem ser indexadas pelos mecanismos de buscas de sites como o Google, Yahoo, Bing (já vimos sobre em outro artigo).

Este arquivo deve ser analisado e seus links acessados, pois esse arquivo para alguns é uma forma de esconder informações dos servidores de busca garantindo a segurança por obscuridade, se ninguém ver ninguém sabe. Porém nas mãos de um atacante este arquivo é uma rica fonte de informações e contraproducente na perspectiva da segurança da informação, já que a intenção é esconder links do mecanismo de busca, mas o arquivo está no servidor web de forma pública.

O web spidering não irá dar uma visão total da aplicação web, pois ela tem algumas dificuldades na hora de colher as informações de forma automática das aplicações web, tais como:

* Não analisa manipulações avançadas em JavaScript;
* Links compilados do lado do cliente como Flash ou Applets Java não são pegos pelo Spidering;
* Alguns Forms HTML fazem validações para passar para o próximo estágio, e as ferramentas automáticas não podem prever essas validações causando erros na resposta do servidor web ou aplicações web;
* Parecido com os Forms HTML, existem páginas que se comportam de forma diferente a cada acesso não sendo possível passar automaticamente essas variações. Por exemplo, em uma página de *accounts.aspx*, ela pode ter a opção *Pessoa Física* e *Pessoa Jurídica*, e daí se comportar de forma diferente para cada escolha;
* Páginas web com áreas de dados voláteis, tais como times e números randômicos;
* Quando a aplicação web exige usuário e senha, e para isso, temos um acesso básico com um usuário, o web spidering pode não ser interessante pelas seguintes razões:
  + No spidering ele pode solicitar o logout e fazer uma análise de forma *“deslogada”*;
  + No spidering pode ser submetido um valor para uma função sensível causando o termino da sessão;
  + Se a aplicação web usar um token para cada página acessada, o spidering pode causar diversos erros pela falta do token e o termino da sessão pode acontecer após um certo número elevado de requisições inválidas;
  + A aplicação web pode ter filtros de browsers clientes que são permitidos (User-Agent), bloqueando todas as requisições do web spidering automático.

**Observação:** Em algumas aplicações web, o uso do web spidering automático pode gerar problemas sérios. Por exemplo, pode ser deletado usuários, derrubar a aplicação ou o banco de dados, reiniciar o servidor web entre outros problemas. Por isso, cuidado.

A forma mais sofisticada e controlada de se fazer o web spidering é de forma manual. Aqui você irá navegar normalmente pela aplicação web utilizando o browser e analisar todas as funções da aplicação. Podemos combinar a interceptação por um proxy e o spidering monitorando todas as requisições e respostas. Para fazermos o web spidering podemos utilizar o Burp Suite e o WebScarab.

Os benefícios de um web spidering manual são:

* Toda a complexidade da aplicação web pode ser monitorada em tempo de acesso pelo proxy/spider;
* Todas as submissões de dados podem ser preenchidas e validadas, ou manipuladas acompanhando as respostas pela aplicação web;
* Mesmo fazendo o spidering em uma aplicação web em que estamos “logado”, se a aplicação nos “deslogar”, por causa do teste feito, você poderá analisar o comportamento da aplicação e voltar para a tela de login, logar e continuar seu spidering;
* Não correrá o risco de apagar usuários da aplicação que não podem ser apagados. Além disso, você poderá tentar criar usuário e apagar esse usuário criado, para analisar as permissões que temos no sistema.

**1 – Sipdering Automático com Burp Suite**

Para fazermos o Web Spidering:

1. Configure seu browser para usar o Burp Suite ou WebScarab como proxy local;
2. Navegue normalmente pela aplicação, acesse todos os links/URLs. Submeta todos os dados de todos os Forms HTML. Faça a navegação com o JavaScript habilitado e desabilitado, assim como, com o Cookie habilitado e desabilitado para analisar o comportamento da aplicação web. Utilize várias configurações de browser (User-Agent) para ver se é habilitado outras funções.
3. Reavalie o sitemap gerado pelo proxy/spider para identificar pontos que você deixou passar e refaça os testes, este é um processo que você deve fazer várias vezes.
4. Oportunamente, enumere o que você já achou e descarte os pontos que não temos interesse ou é irrelevante.

Vamos abrir o Burp Suite e configurar o proxy do Browser para passar pelo Burp Suite.

Com o Firefox aberto, clique em **Menu🡪Preferences🡪Advanced** em **Connection** clique no botão **Settings**.

Em **Connection Settings**, selecione **Manual proxy configuration**. Em **HTTP Proxy** digite o endereço do proxy, que no nosso caso o Burp Suite está configurado para ouvir em **127.0.0.1** na porta **8080/TCP**. Após isso, selecione o checkbox **Use this proxy server for all protocols** e clique em **OK**.

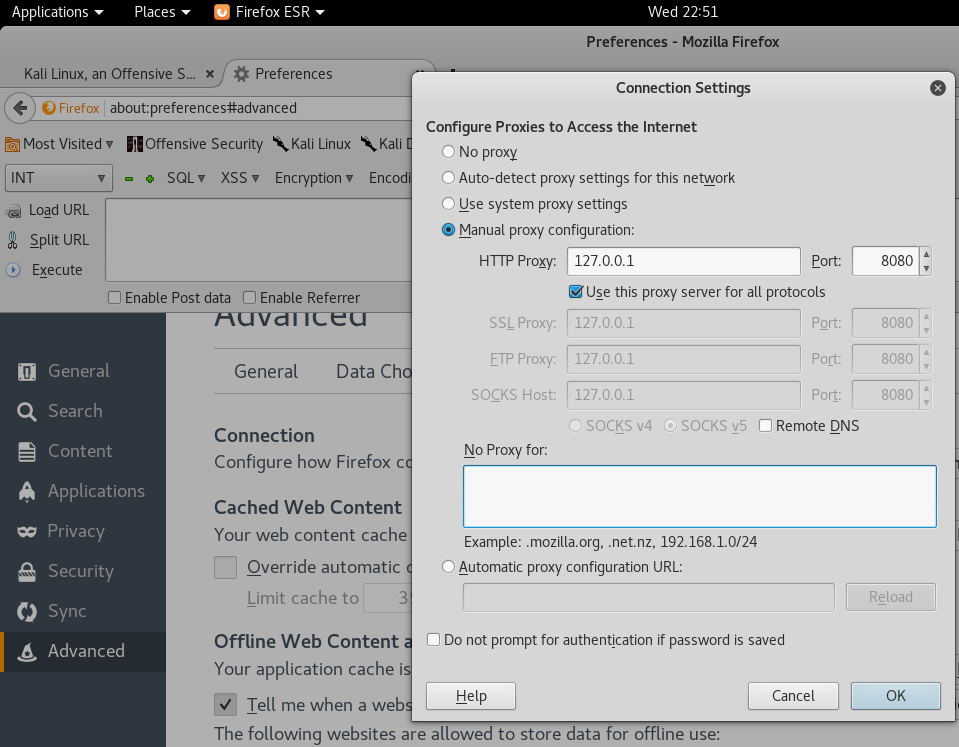


Figura 01 – Configurando o Browser para Proxiar Pelo Burp Suite

Agora vá para o Burp Suite, clique na aba **Proxy**, depois clique na sub-aba **Options** para ver as opções do **Burp Proxy**. Certifique-se que o Burp está configurado para escutar no endereço de loopback e na porta **8080/TCP**, caso não esteja configure ele para trabalhar com essas opções.

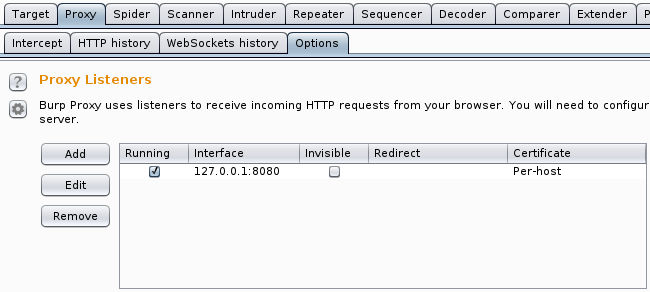


Figura 02 – Opções do Burp Proxy

Agora clique na sub-aba **Intercept**, E observe se o botão **Intercept** está desligado (**intercept is off**), pois não queremos que ele pause a comunicação, nossa intenção agora é apenas mapear a aplicação e não manipular.

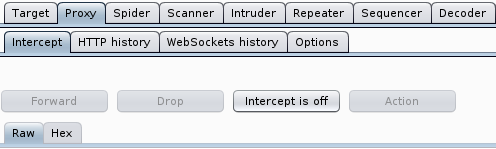


Figura 03 – Burp Proxy – Intercept is Off

Agora podemos carregar a página que gostaríamos de fazer o crawling, no nosso caso o site da Alturo Mutual (https://www.altoromutual.com).

Em seguida vamos dar uma olhada na aba **Target** do Burp Suite para ver se o website está sendo mapeado.

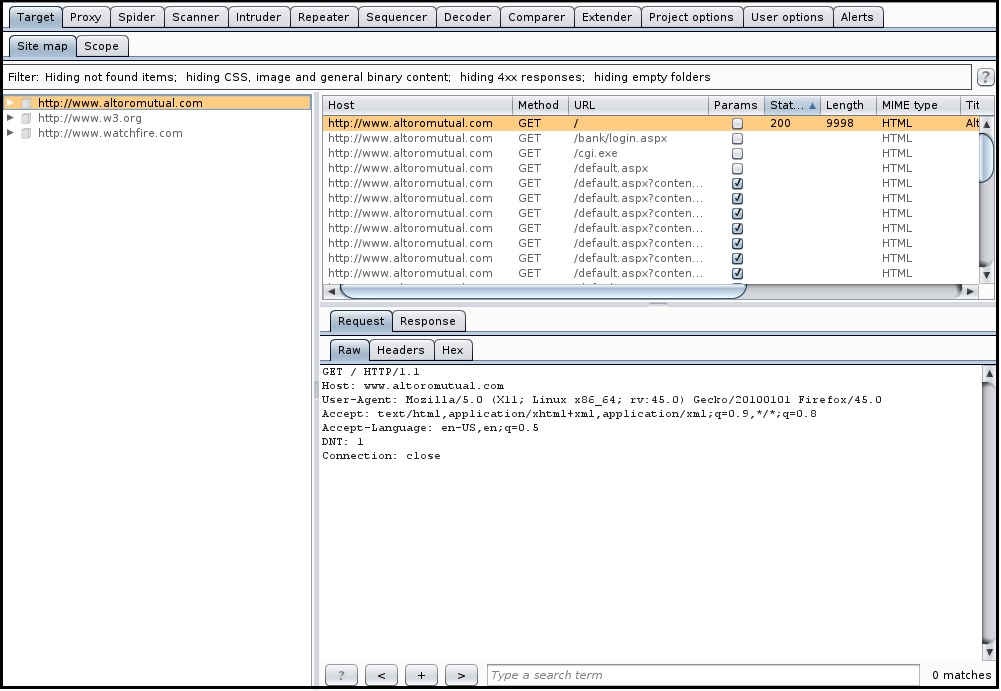


Figura 04 – Crawling do WebSite da Altoro Mutual

Como podemos observar, podemos acompanhar as requisições e as respostas pelas respectivas abas **Request** e **Response**. Além disso, podemos ver os dados em modo puro (**Raw**), o cabeçalho (**Headers**) e em formato hexadecimal (**Hex**).

Todos os links são relacionados ao link principal. Quanto mais complexo for o WebSite mais links aparecerão para serem analisados.

Podemos clicar com o botão direito do mouse no links, ou diretório para copiarmos ou fazer um spidering.

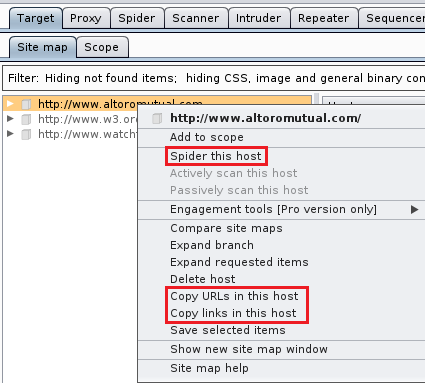


Figura 05 – Copiar ou Fazer Spidering

Para fazer um Spider Crawling automático, basta selecionamos **Spider this host**. Lembre-se que esse tipo de Spider pode trazer prejuízos ao nosso teste, mas as vezes é a nossa única opção devido prazos.

Podemos ver o progresso do Spidering na aba **Spider** do Burp Spider. E acompanhar os retornos na aba **Target**.

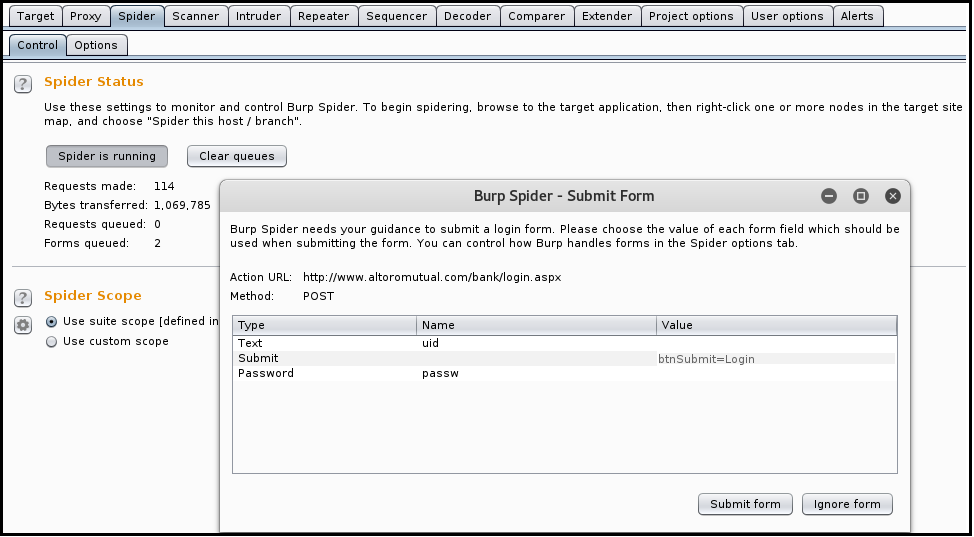


Figura 06 – Spidering Automatizado no Burp Spider

Podemos salvar nosso spidering automático em formato XML clicando com o botão direito do mouse sobre o link apontando para **Save selected items**. Em seguida de um nome para o arquivo e clique no botão **Save**.

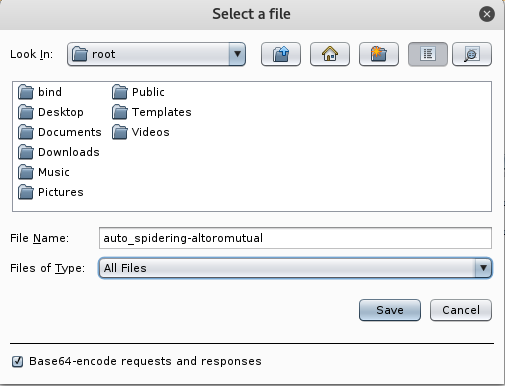


Figura 07 – Salvando o Resultado do Spidering

Observe que está selecionado que as requisições e respostas estarão encodados em Base64.

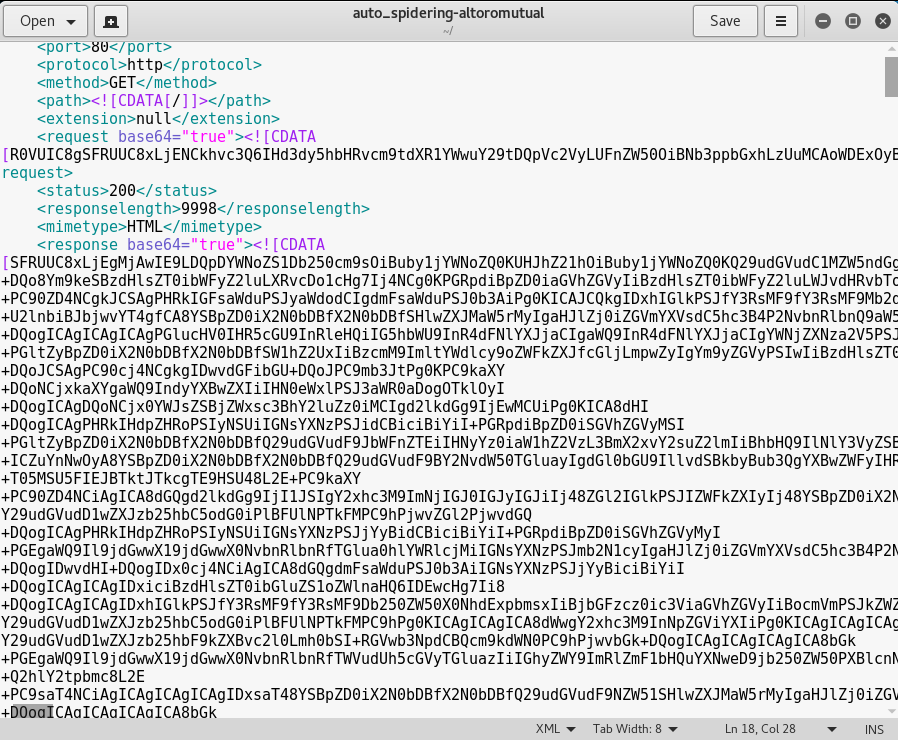


Figura 08 – Arquivo de Resultado do Spidering Automático

**2 – Sipdering Manual com Burp Suite**

A forma de enumerar o conteúdo e funções das aplicações web é de forma manual, navegando com o browser enquanto está configurado com proxy local.

Podemos começar navegando na página inicial da aplicação web e ir seguindo todos os links encontrados na aplicação, além de navegar por todas as funções (tais como logins de acesso, reset de senha, recuperação de usuários e senha, etc.). Se a aplicação tiver sitemap podemos utilizá-lo para enumerar o conteúdo.

Desta forma iremos mapear todos os inputs que a aplicação web tem, para assim testarmos essas entradas.

Para fazermos o Spirering manual basta proxiamos nossa navegação como no automático porém sem mandar fazer o Spider.

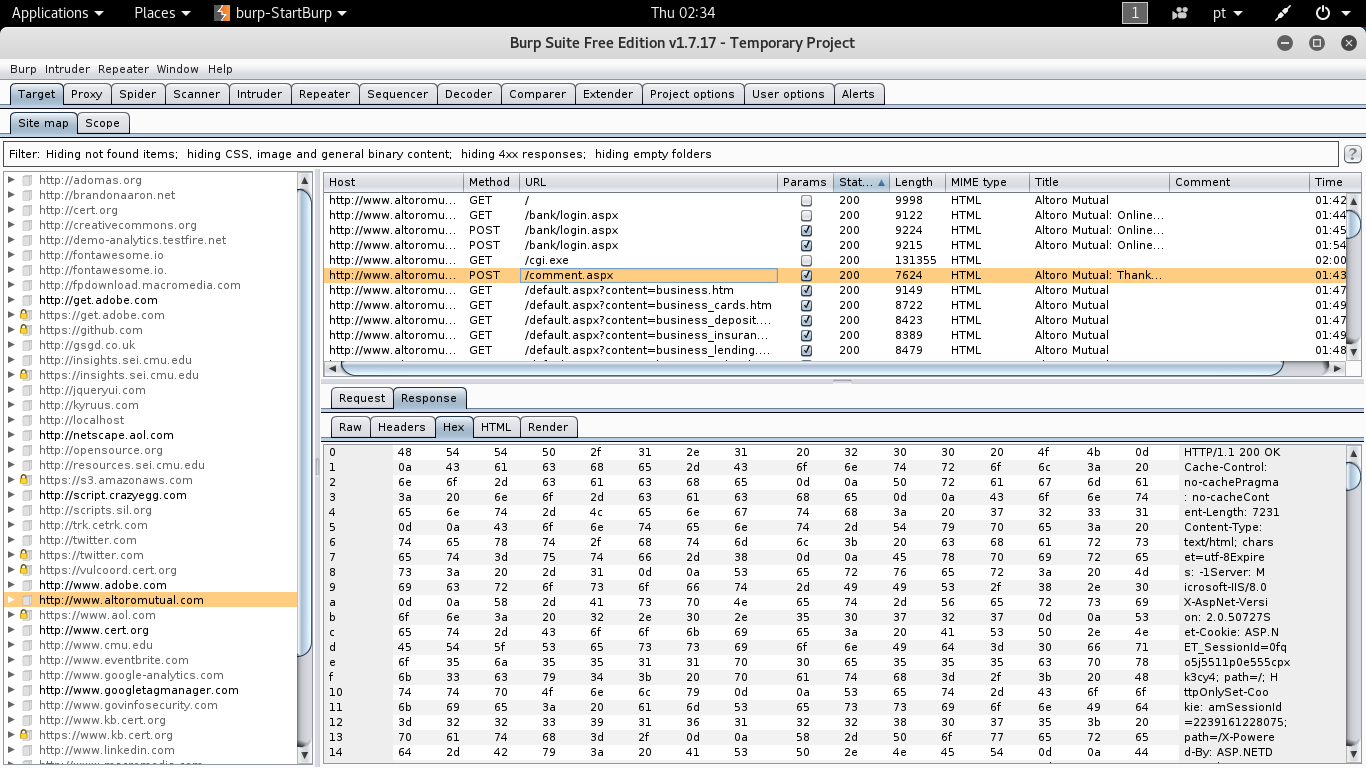


Figura 09 – Spider Crawling Manual

Vamos salvar nosso Spidering Manual para posteriores consultas.

**3 – Descobrindo Conteúdo Oculto na Aplicação Web**

É comum nas aplicações web existirem funções ou arquivos soltos que não tenha um link. Um bom exemplo, são funções que habilitam o debug na aplicação através de uma variável e não foi removido no ambiente de produção.

Outro exemplo são funções que são habilitadas dependendo do seu acesso à aplicação web (anônimo, autenticado ou administrador).

Outros casos interessantes são os apontados abaixo:

* Cópias de backup da aplicação web na raiz do servidor web, contendo todo o código fonte da aplicação;
* Backup de arquivos contendo dados sensíveis, como dump do banco de dados ou sua estrutura;
* Novas funcionalidades que ainda não está linkado com a aplicação principal;
* Arquivos com funcionalidades padrão da aplicação web que não foram removidas, tais como os CMS (Content Management System);
* Versões antigas de arquivos que não foram removidos do servidor antigo que foram corrigidos na nova versão;
* Arquivos contendo credenciais e configurações;
* Arquivos de código fonte usado em aplicações compilada;
* Comentários no código fonte informando funções de teste;
* Arquivos de log contendo dados sensíveis, como nome de usuários validos, tokens de sessões e URLs visitadas.

Para descobrirmos diversos caminhos e arquivos ocultos, podemos utilizar a técnica de brute-force. Para isso, vamos crie um arquivo (**wordlist\_urls.txt**) e alimente esse arquivo com as URLs e arquivos que gostaríamos de encontrar.

Em um brute-force devemos nos atentar no **Status Code** e no **Length** (Tamanho) dos pacotes, pois são eles que sinaliza que houve uma anomalia na busca, indicando que o caminho foi encontrado ou uma ação foi tomada.

Uma resposta HTTP com o **Status 302 (Found)** para um arquivo dentro da busca, indica que foi encontrado o arquivo e tem um redirecionamento para o login, pois o acesso é restrito.

Vale lembrar que a resposta **200 OK** não indica que existe o arquivo, pois o desenvolvedor pode personalizar a mensagem de erro, assim como uma resposta **404 Not Found**, informando que não existe o arquivo.

Na busca por arquivos ocultos devemos adicionar na busca extensões de arquivos que possa conter dados interessantes tais como: ***.txt, .bak, .src, .inc, .java, .cs, .c, .cpp, .tmp, .php~1, .tar, .gzip, .zip, .bz2, .sql***, etc.

**Observação:** No Recon Passive nós lidamos com os arquivos ***robots.txt*** eles são uma boa base para alimentar nosso brute-force.

**4 – Descobrindo Parâmetros Oculto na Aplicação Web**

Use uma lista de parâmetros comuns para opções booleanas como debug=(true, false, yes, no, 1, 0). Sempre analisamos essas opções nas URLs e no corpo da mensagem HTTP.

Monitore todas as respostas do servidor para identificar anomalias ao passar os valores aos parâmetros booleanos.

Procure pelos parâmetros em campos de debug, login, procura/pesquisa e upload/download de arquivos.

**5 – Identificando Pontos de Entradas**

Devemos observar todas as possibilidades de entrada na aplicação web em que um usuário pode enviar para a aplicação. Dentre elas podemos destacar:

* Todas URLs com valores passados nos parâmetros na URL pelo método GET;
* Todos os parâmetros submetidos na URL;
* Todos os parâmetros submetidos no corpo da mensagem HTTP pelo método POST;
* Todos os Cookies;
* Todos os campos do header HTTP, em particular as informações enviadas através dos campos User-Agent, Referer, X-Forwarded-For, Accept, Accept-Length e Host.

Os campos User-Agent e Referer são pontos de entradas mais promissores para ataques.

Algumas aplicações executam processos adicionais através do campo Referer. Por exemplo, alguns aplicativos web podem processar as informações do campo Referer para saber que o usuário chegou a sua aplicação através de um mecanismo de busca, e desta forma personalizar a resposta, ou ainda, contabilizar quantos acessos foram feitos por meio do mecanismo de busca. De qualquer forma podemos injetar conteúdo neste campo e observar a resposta.

O campo User-Agent também é processado em algumas aplicações web, pois hoje em dia temos os mais diversos clientes web (browsers) acessando as aplicações web através de dispositivos como: smartphone, tablete, etc.

Sendo assim, algumas aplicações personaliza as páginas dependendo do dispositivo e das limitações dos browsers clientes.

Outro campo interessante é o X-Forwarded-For utilizado por aplicativos de balanceamento de carga (Load Balance), podemos injetar SQLi ou XSS.

Outra forma de enviarmos dados de entrada não convencional, são os serviços hospedados em outros servidores como SMTP, enviando e-mail através de Form HTML, ou API e SOAP. Chamamos essa forma de abuso de Out-of-Band Channels.

Devemos observar as URLs que mascaram os parâmetros, enviando apenas os valores dos parâmetros como se fossem diretórios. Por exemplo, aplicativos que utilizam REST-style URL. Uma URL típica utilizando REST-style URL:

http://dominio-alvo.com/electronic/memory/notebook

Pode ser interpretada da seguinte forma:

http://dominio-alvo.com/shop.php?type=electronic&prod=memory&cat=notebook

Além disso, temos aplicações que não seguem o formato padrão de nome=valor nos parâmetros. Vamos ver alguns exemplos encontrados por aí:

* /dir/file;prod=1&cod=2
* /dir/file?prod=1$cod=2
* /dir/file/prod%3D1%26cod%3D2
* /dir/1.2/file
* /dir/1=2/file
* /dir/file?param=1:2
* /dir/file?data%3Cprod%3E1%3C%2Fcod%3E2%3C%2F

Mesmo não utilizando os formatos padrões de aplicações web, não deixa de ter as vulnerabilidades comuns as aplicações web.

**6 – Identificando as Tecnologias Utilizada do Lado do Servidor Web**

Podemos identificar quais tecnologias estão envolvidas na publicação das aplicações web.

Uma das formas de descobrimos qual o servidor web utilizado é através do **banner grabbing**, esta informação está no campo **Server** do header HTTP. Porém essa informação pode ser alterada pelo administrador do servidor web, por isso, ela não é muito confiável e deve ser correlacionada com outras informações tais como as que se seguem abaixo.

Extensões dos arquivos no servidor web:

* **asp** 🡪 Microsoft Active Server Page;
* **aspx** 🡪 Microsoft ASP.NET;
* **jsp** 🡪 Java Server Page;
* **cfm** 🡪 Cold Fussion;
* **php** 🡪 Linguagem PHP;
* **d2w** 🡪 WebSphere;
* **pl** 🡪 Linguagem Perl;
* **py** 🡪 Linguagem Python;
* **dll** 🡪 Usualmente compilado com C ou C++;
* **nsf** ou **ntf** 🡪 Lotus Domain.

Temos os erros padrões que costumam ter características diferentes dependendo do servidor web

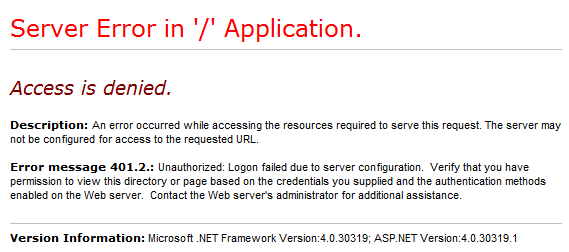


Figura 01 – Erro Padrão do Microsoft IIS e Framework .NET

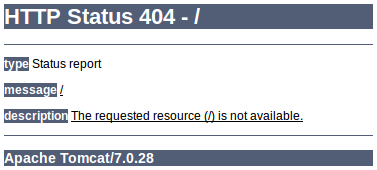


Figura 02 – Erro Padrão do Apache Tomcat

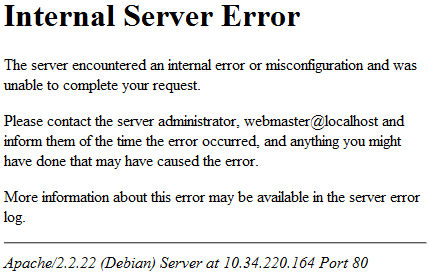


Figura 03 – Erro Padrão do Apache

Outro indicio de qual servidor web suporta as aplicações web são os diretórios encontrado.

* **Serlet** 🡪 Java Servelets;
* **Pls** 🡪 Oracle Application Server PL/SQL gateway;
* **CFDocs** ou **CFide** 🡪 Cold Fusion;
* **Silverstream** 🡪 Web Server do Silverstream;
* **WebObject** ou **{function}.woa** 🡪 Apple WebObject;
* **Rails** 🡪 Ruby on Rails.

Ainda podemos observe os Tokens de Sessões padrão:

* **JSESSIONID** 🡪 Plataforma Java;
* **ASPSESSIONID** 🡪 Microsoft IIS;
* **ASP.NET\_SessionID** 🡪 Microsoft ASP.NET;
* **CFID/CFTOKEN** 🡪 Cold Fusion;
* **PHPSESSID** 🡪 PHP.

**7 – Dissecando as Requisições**

Considerando a seguinte URL:

http://dominio-alvo.com/calend.php?name=new%20application&isExpired=0&startDate=22%2F09%2F2016&endDate=22%2F10%2F2016&OrderBy=name

Primeira coisa a se observar é que se trata de uma aplicação em PHP (calend.php) pois a extensão da aplicação é .php. Outro detalhe interessante é que o parâmetro OrderBy é tipicamente utilizado para indexar base de dados. Com isso, podemos deduzir que o back-end é uma pesquisa SQL, e podemos testar um SQLi neste parâmetro. Também temos o parâmetro isExpired, que é um parâmetro booleano que informa se o conteúdo expira, podemos subverter esse valor colocando o valor 1 e vendo o comportamento da aplicação.

http://dominio-alvo.com/work.aspx?template=NewBrantch.tpl&loc=/default&ver=2.1&edit=false

Semelhante ao link anterior, podemos observar a tecnologia utilizada que é ASP.NET (.aspx) e que chama um template, localizado no diretório */default*. Aqui temos um possível valor de localização que pode ser testado um transversal. Outro parâmetro interessante o edit=false indicando ser um parâmetro booleano, onde podemos trocar de false para true. O parâmetro *ver* pode ser mudado para testar se ele chama uma versão de template antiga e que tenha alguma vulnerabilidade para ser explorada.

POST /feedback.php HTTP/1.1

Host: domínio-alvo.com

Content-Length: 308

from=well@how2security.com.br&tosupport@dominio-alvo.com&subject=Problema+de+acesso&message=Por+favor,+me\_ajude...

Como podemos observar os parâmetros se trata de envio de e-mail, com isso, podemos testar se o SMTP Relay está habilitado para esse servidor.

**8 – Referências Bibliográficas**

**[1]** MUNIZ, JOSEPH and LAKHANI, AAMIR – Web Penetration Testing with Kali Linux, 2013, Birmingham – Mumbai, Packet Publishing.

**[2]** GRAVES, KIMBERLY – CEH – Official Ethical Hacker Review Guide, Indianapolis - Indiana, 2007, Wiley Publishing Inc.

**[3]** STUTTARD, DAFYDD and PINTO, MARCUS – The Web Application Hacker’s Handbook, Indianapolis - Indiana, 2º Edition, Wiley Publishing Inc.

**[4]** SCAMBRAY, JOEL, McCLURE, STUART and KURTZ, GEORGE – Hackers Expostos Segredos e Soluções para a Segurança de Redes, São Paulo, 2001, MAKRON Books Ltda.

**[5]** LONG, JOHNNY - GOOGLE HACKING - Para Teste de Invasão [tradução Sérgio Pereira Couto] 2005, DIGERATI (BOOKS).

**[6]** MELO, Sandro – Estudo de Técnicas para Exploração de Vulnerabilidades em Redes TCP/IP, 2º Ed, Rio de Janeiro, 2006, Editora Alta Books Ltda.

**[7]** AHARONI, Mati; Offensive Security Lab Exercises - Curso de Pen-Test oferecido pela Offensive Security, 2007.

**[8]** SCAMBRAY, Joel; LIU, Vicent; SIMA, Caleb - Hacking Exposed Web Application: Web Application Security Secrets And Solutions – 3ª Edition, New York – 2011, McGraw-Hill.

**[9]** DHANJANI, Nitesh; RIOS, Billy; HARDIN, Brett - Hacking A Próxima Geração – Rio de Janeiro – 2011, Alta Books.

**[10]** HOPE, Paco; WALTHER, Ben – Web Segura Guia de Testes e Soluções – Rio de Janeiro – 2009, Alta Books.