Building A Module a11y.text Building A Module Writing your first Metasploit module can be a daunting task, especially if one does not code in Ruby on a regular basis. Fortunately the languageâ€™s syntax is intuitive enough, for anyone with prior programming and scripting knowledge, to make the transition (from Python for example) to Ruby. Before taking the plunge into module construction and development, letâ€™s take a quick look at the some of the modules currently in place. These files can be used as our base for re-creating an attack on several different supported protocols, or crafting ones own custom module.

root@kali:/usr/share/metasploit-framework/lib/msf/core/exploit# ls

afp.rb dect_coa.rb mixins.rb smb

arkeia.rb dhcp.rb mssql_commands.rb smb.rb

browser_autopwn.rb dialup.rb mssql.rb smtp_deliver.rb

brute.rb egghunter.rb mssql_sqli.rb smtp.rb

brutetargets.rb exe.rb mysql.rb snmp.rb

capture.rb file_dropper.rb ndmp.rb sunrpc.rb

cmdstager_bourne.rb fileformat.rb ntlm.rb tcp.rb

cmdstager_debug_asm.rb fmtstr.rb omelet.rb telnet.rb

cmdstager_debug_write.rb ftp.rb oracle.rb tftp.rb

cmdstager_echo.rb ftpserver.rb pdf_parse.rb tns.rb

cmdstager_printf.rb http pdf.rb udp.rb

cmdstager.rb imap.rb php_exe.rb vim_soap.rb

cmdstager_tftp.rb ip.rb pop2.rb wbemexec.rb

cmdstager_vbs_adodb.rb ipv6.rb postgres.rb wdbrpc_client.rb

cmdstager_vbs.rb java.rb powershell.rb wdbrpc.rb

db2.rb kernel_mode.rb realport.rb web.rb

dcerpc_epm.rb local remote winrm.rb

dcerpc_lsa.rb local.rb riff.rb

dcerpc_mgmt.rb lorcon2.rb ropdb.rb

dcerpc.rb lorcon.rb seh.rb Here we see several modules of interest, such as prepackaged protocols for Microsoftâ€™s SQL, HTTP, TCP, FTP, SMTP, SNMP, Oracle, and many more. These files undergo constant changes and updates, adding new functionalities over time. Letâ€™s start with a very simple program, navigate to /usr/share/metasploit-framework/modules/auxiliary/scanner/mssql and create the required Metasploit folder structure under your home directory to store your custom module. Metasploit automatically looks in this folder structure so no extra steps are required for your module to be found. root@kali:/usr/share/metasploit-framework/modules/auxiliary/scanner/mssql# mkdir -p ~/.msf4/modules/auxiliary/scanner/mssql Then do a quick cp mssql_ping.rb ~/.msf4/modules/auxiliary/scanner/mssql/ihaz_sql.rb

root@kali:/usr/share/metasploit-framework/modules/auxiliary/scanner/mssql# cp mssql_ping.rb ~/.msf4/modules/auxiliary/scanner/mssql/ihaz_sql.rb Open the newly-created file using your favourite editor and weâ€™ll begin crafting our example module, walking through each line and what it means: ##

$Id: ihaz_sql.rb 7243 2009-12-04 21:13:15Z rel1k $ >--- automatically gets set for us when we check in

##

##

This file is part of the Metasploit Framework and may be subject to >---- licensing agreement, keep standard

redistribution and commercial restrictions. Please see the Metasploit

Framework web site for more information on licensing and terms of use.

http://metasploit.com/framework/

##

require 'msf/core' >--- use the msf core library

class MetasploitModule < Msf::Auxiliary >---- its going to be an auxiliary module

include Msf::Exploit::Remote::MSSQL   >----- we are using remote MSSQL right?

include Msf::Auxiliary::Scanner >---------- it use to be a SQL scanner

def initialize >---- initialize the main section

super(

'Name' => 'I HAZ SQL Utility',   >------- name of the exploit

'Version' => '$Revision: 7243

Now that you have a basic idea of the module, save the above code (without the >â€”â€” comment strings) and letâ€™s run it in msfconsole. msf > search ihaz

[*] Searching loaded modules for pattern â€˜ihazâ€™â€¦ Auxiliary a11y.text Auxiliary Name Description scanner/mssql/ihaz_sql MSSQL Ping Utility msf > use scanner/mssql/ihaz_sql

msf auxiliary(ihaz_sql) > show options Module options: Name Current Setting Required Description HEX2BINARY /pentest/exploits/framework3/data/exploits/mssql/h2b no The path to the hex2binary script on the disk

MSSQL_PASS no The password for the specified username

MSSQL_USER sa no The username to authenticate as

RHOSTS yes The target address range or CIDR identifier

THREADS 1 yes The number of concurrent threads msf auxiliary(ihaz_sql) > set RHOSTS doesntmatter

RHOSTS => doesntmatter

msf auxiliary(ihaz_sql) > exploit

I HAZ SQL!!!! [ ] Scanned 1 of 1 hosts (100% complete)

[ ] Auxiliary module execution completed Success! Our module has been added! Now that we have a basic understanding of how to add a module, letâ€™s take a closer look at the MSSQL module written for the Metasploit framework., >------- svn number

'Description' => 'This just prints some funny stuff.', >---------- description of the exploit

'Author' => 'THE AUTHOR', >--- thats you

'License' => MSF_LICENSE >---- keep standard

)

deregister_options('RPORT', 'RHOST') >---- do not specify RPORT or RHOST

end

def run_host(ip) >--- define the main function

begin >---begin the function

puts "I HAZ SQL!!!!" >---- print to screen i haz SQL!!!

end >--- close

end >---- close

end >---- close Now that you have a basic idea of the module, save the above code (without the >â€”â€” comment strings) and letâ€™s run it in msfconsole.

urltomarkdowncodeblockplaceholder40.2319882518606433 Success! Our module has been added! Now that we have a basic understanding of how to add a module, letâ€™s take a closer look at the MSSQL module written for the Metasploit framework. Next Payloads Through MSSQL Prev PHP

Meterpreter