Writing MSF Post Exploitation Module

Automate pwned shell >

/me

- Sanoop Thomas @s4n7h0
- My last talk on MSF was 4 years back
  - https://www.exploit-db.com/docs/26000.pdf
- * not a ruby guy *

/msf architecture

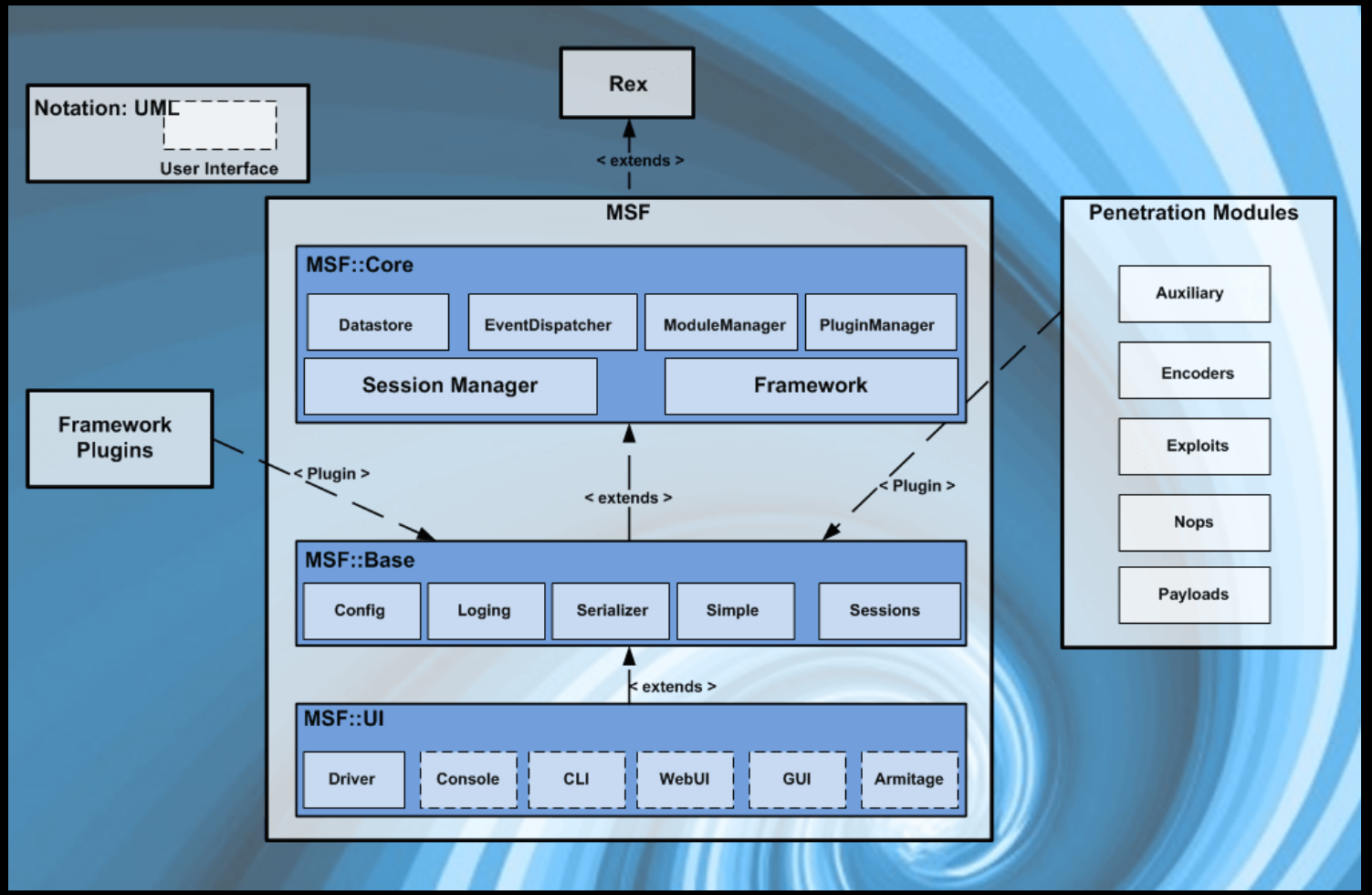

Image Source: https://www.offensive-security.com/metasploit-unleashed/metasploit-architecture

/template

```
##
# This module requires Metasploit: http://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

require 'msf/core'

class MetasploitModule < Msf::Post

  def initialize(info={})
    super(update_info(info,
        'Name'          => '[Platform] [Module Category] [Software] [Function]',
        'Description'   => %q{
          Say something that the user might want to know.
        },
        'License'       => MSF_LICENSE,
        'Author'        => [ 'Name' ],
        'Platform'      => [ 'win', 'linux', 'osx', 'unix', 'bsd' ],
        'Arch'          => [ 'x86' ],
        'SessionTypes'  => [ 'meterpreter', 'shell' ]
    ))
  end

 def run
    # Main method
  end

end
```

/require

```
root@kali:/usr/share/metasploit-framework# cat lib/msf/core/post/windows.rb
# -*- coding: binary -*-

module Msf::Post::Windows
  require 'msf/core/post/windows/error'
  require 'msf/core/post/windows/extapi'
  require 'msf/core/post/windows/accounts'
  require 'msf/core/post/windows/cli_parse'
  require 'msf/core/post/windows/eventlog'
  require 'msf/core/post/windows/file_info'
  require 'msf/core/post/windows/powershell'
  require 'msf/core/post/windows/priv'
  require 'msf/core/post/windows/process'
  require 'msf/core/post/windows/railgun'
  require 'msf/core/post/windows/registry'
  require 'msf/core/post/windows/runas'
  require 'msf/core/post/windows/services'
  require 'msf/core/post/windows/wmic'
  require 'msf/core/post/windows/netapi'
  require 'msf/core/post/windows/shadowcopy'
  require 'msf/core/post/windows/user_profiles'
  require 'msf/core/post/windows/ldap'
  require 'msf/core/post/windows/reflective_dll_injection'
  require 'msf/core/post/windows/kiwi'
  require 'msf/core/post/windows/dotnet'
end
```

/register

```
register_options(
[
  OptString.new('STRING_OPTION', [ true, 'Example for String', 'Hello World' ]),
  OptBool.new('BOOL_OPTION', [ true, 'Example for Boolean', 'TRUE' ]),
  OptInt.new('INTEGER_OPTION', [ true, 'Example for Integer', '0' ]),
  OptAddress.new('ADDRESS_OPTION', [ true, 'Example for IP Address', '' ]),
  OptAddressRange.new('ADDRESSRANGE_OPTION', [ false, 'Example for Address Range', '' ]),
  OptPort.new('PORT_OPTION', [ true, 'EXample for PORT', '445' ]),
  OptPath.new('PATH_OPTION',  [ false, "Example for Path", 'c:\\test'])
], self.class)
```

```
Basic options:
  Name                Current Setting  Required  Description
  ----                ---------------  --------  -----------
  ADDRESSRANGE_OPTION                  no        Example for Address Range
  ADDRESS_OPTION                       yes       Example for IP Address
  BOOL_OPTION         TRUE             yes       Example for Boolean
  INTEGER_OPTION      0                yes       Example for Integer
  PATH_OPTION         c:\test          no        Example for Path
  PORT_OPTION         445              yes       EXample for PORT
  SESSION                              yes       The session to run this module on.
  STRING_OPTION       Hello World      yes       Example for String
```

/advanced options

```
register_options(
[
  OptString.new('PATH',  [ true, 'File path on remote system', 'C:\\msfdemo\\example.txt'])
], self.class)

register_advanced_options(
[
    OptString.new('ANOTHER_PATH', [ false, 'Another file path ', '' ])
], self.class)
```

```
msf post(example1) > show options

Module options (post/windows/example/example1):

   Name     Current Setting         Required  Description
   ----     ---------------         --------  -----------
   PATH     C:\msfdemo\example.txt  yes       File path on remote system
   SESSION                          yes       The session to run this module on.

msf post(example1) > show advanced

Module advanced options (post/windows/example/example1):

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   ANOTHER_PATH                   no        Another file path
   VERBOSE       false            no        Enable detailed status messages
   WORKSPACE                      no        Specify the workspace for this module
```

/access datastore and data_directory

- `variable_name = datastore['OPT_VARIABLE']`
- `Msf::Config.data_directory`

```
root@kali:/usr/share/metasploit-framework# ls data/
SqlClrPayload          flash_detector  markdown_doc  post              webcam
cpuinfo                ipwn            meterpreter   snmp              wmap
eicar.com              isight.bundle   mime.yml      sounds            wordlists
eicar.txt              john.conf       msfcrawler    templates
emailer_config.yaml    lab             passivex      vncdll.x64.dll
exploits               logos           php           vncdll.x86.dll
```

/run

```
def run
    print_line('let\'s say Hello World')
    print_status('well, status text')
    print_good('Great, Good text')
    print_error('Oops!! Error text')
end
```

```
msf post(test_module) > run

let's say Hello World
[*] well, status text
[+] Great, Good text
[-] Oops!! Error text
[*] Post module execution completed
```

/readfile

```
  register_options(
  [
    OptString.new('PATH',  [ true, 'File path on remote system', 'C:\\msfdemo\\example.txt'])
  ], self.class)

end

def run

 file_path = datastore['PATH']

 print_status('Starting module...')
 print_line('')

 if exist?(file_path)
   file_contents = read_file(file_path)
   print_good('File contents:')
   print_line(file_contents)
 else
   print_error('Cannot read specified file!')
 end

end
```

/systeminfo

```
def run

 environ = session.sys.config.getenvs('COMPUTERNAME')
 print_good("OS: #{session.sys.config.sysinfo['OS']}")
 print_good("Computer name: #{environ['COMPUTERNAME']}")
 print_good("Current User: #{session.sys.config.getuid}")
 print_line('')

end
```

/cmd_exec

```
def run

  cmdexec = datastore['CMDEXEC']
  print_status("Executing command: #{cmdexec}")
  print_line('')

  output = cmd_exec(cmdexec)
  print_line(output)
  print_line('')

end
```

/railgun

- **client.railgun**.DLL.Function(parameter)
- Example,
  - client.railgun.user32.MessageBoxA(0, "Hello World", "MSF talking", "MB_OKCANCEL")

Thanks

Shell is just the beginning