Sign in

Hackplayers / evil-winrm Public

Notifications Fork 609 Star 4.5k

Code Pull requests Actions Security Insights

evil-winrm / evil-winrm.rb

Executable File · 1029 lines (910 loc) · 120 KB

Code Blame Raw

```ruby
#!/usr/bin/env ruby
# frozen_string_literal: true

# Author: CyberVaca
# Twitter: https://twitter.com/CyberVaca_
# Based on the Alamot's original code

# Dependencies
require 'English'
require 'winrm'
require 'winrm-fs'
require 'stringio'
require 'base64'
require 'readline'
require 'optionparser'
require 'io/console'
require 'time'
require 'fileutils'
require 'logger'

# Constants

# Version
VERSION = '3.5'

# Msg types
```

```ruby
TYPE_INFO = 0
TYPE_ERROR = 1
TYPE_WARNING = 2
TYPE_DATA = 3
TYPE_SUCCESS = 4

# Global vars

# Available commands
$LIST = %w[Bypass-4MSI services upload download menu exit]
$COMMANDS = $LIST.dup
$CMDS = $COMMANDS.clone
$LISTASSEM = [''].sort
$DONUTPARAM1 = ['-process_id']
$DONUTPARAM2 = ['-donutfile']

# Colors and path completion
$colors_enabled = true
$check_rpath_completion = true

# Path for ps1 scripts and exec files
$scripts_path = ''
$executables_path = ''

# Connection vars initialization
$host = ''
$port = '5985'
$user = ''
$password = ''
$url = 'wsman'
$default_service = 'HTTP'
$full_logging_path = "#{Dir.home}/evil-winrm-logs"

# Redefine download method from winrm-fs
module WinRM
  module FS
    class FileManager
      def download(remote_path, local_path, chunk_size = 1024 * 1024, first = true, size: -1)
        @logger.debug("downloading: #{remote_path} -> #{local_path} #{chunk_size}")
        index = 0
        return download_dir(remote_path, local_path, chunk_size, false) if remote_path.match?(/(\*\
        output = _output_from_file(remote_path, chunk_size, index)
        return download_dir(remote_path, local_path, chunk_size, true) if output.exitcode == 2
        return false if output.exitcode >= 1

        File.open(local_path, 'wb') do |fd|
          begin
            out = _write_file(fd, output)
            index += out.length
            until out.empty?
              yield index, size if size != -1
              output = _output_from_file(remote_path, chunk_size, index)
              return false if output.exitcode >= 1

              out = _write_file(fd, output)
              index += out.length
            end
          rescue EstandardError => err
            @logger.debug("IO Failed: " + err.to_s)
            raise
          end
        end
      end

      def download_dir(remote_path, local_path, chunk_size, first)
        index_exp = remote_path.index(/(\*\.|\*\*|\.\*|\*)/) || 0
        remote_file_path = remote_path

        if index_exp > 0
          index_last_folder = remote_file_path.rindex(/[\\\/]/, index_exp)
          remote_file_path = remote_file_path[0..index_last_folder-1]
        end

        FileUtils.mkdir_p(local_path) unless File.directory?(local_path)
        command = "Get-ChildItem #{remote_path} | Select-Object Name"

        @connection.shell(:powershell) { |e| e.run(command) }.stdout.strip.split(/\n/).drop(2).each
          download(File.join(remote_file_path.to_s, file.strip), File.join(local_path, file.strip),
        end
      end

      true
    end
  end
end

# Class creation
class EvilWinRM
  # Initialization
  def initialize
    @psLoaded = false
    @directories = {}
```

```
      @directories = {}
```

```
          is_valid = current_vals['time'] > current_time - @cache_ttl
          result = current_vals['files'] if is_valid
          @directories.delete(a_path) unless is_valid
        end

        result
      end

      def set_cache(n_path, paths)
        return if n_path.nil? || n_path.empty?

        a_path = normalize_path(n_path)
        current_time = Time.now.to_i
        @directories[a_path] = { 'time' => current_time, 'files' => paths }
      end

      def normalize_path(str)
        Regexp.escape(str.to_s.gsub('\\', '/'))
      end

      def get_dir_parts(n_path)
        return [n_path, ''] unless (n_path[-1] =~ %r{/$}).nil?

        i_last = n_path.rindex('/')
        return ['./', n_path] if i_last.nil?

        next_i = i_last + 1
        amount = n_path.length - next_i

        [n_path[0, i_last + 1], n_path[next_i, amount]]
      end

      def complete_path(str, shell)
```

```ruby
      return unless @completion_enabled
      return unless !str.empty? && !(str =~ %r{^(\./|[a-z,A-Z]:|\.\./|~/|/)*}i).nil?

      n_path = str
      parts = get_dir_parts(n_path)
      dir_p = parts[0]
      nam_p = parts[1]
      result = []
      result = get_from_cache(dir_p) unless dir_p =~ %r{^(\./|\.\./|~|/)}

      if result.nil? || result.empty?
        target_dir = dir_p
        pscmd = "$a=@();$(ls '#{target_dir}*' -ErrorAction SilentlyContinue -Force |Foreach-Object {

        output = shell.run(pscmd).output
        s = output.to_s.gsub(/\r/, '').split(/\n/)

        dir_p = s.pop
        set_cache(dir_p, s)
        result = s
      end
      dir_p += '/' unless dir_p[-1] == '/'
      path_grep = normalize_path(dir_p + nam_p)
      path_grep = path_grep.chop if !path_grep.empty? && path_grep[0] == '"'
      filtered = result.grep(/^#{path_grep}/i)
      filtered.collect { |x| "\"#{x}\"" }
    end
  end

  # Class to create array (tokenize) from a string
  class String
    def tokenize
      split(/\s(?=(?:[^'"]|'[^']*'|"[^"]*")*$)/)
        .reject(&:empty?)
        .map { |s| s.gsub(/(^ +)|( +$)|(^["']+)|(["']+$)/, '') }
    end
  end

  # Execution
  e = EvilWinRM.new
  e.main
```