Sign in

rapid7 / metasploit-framework Public

Notifications Fork 14k Star 34.1k

Code Issues 411 Pull requests 44 Discussions Actions Projects 1 Wiki

metasploit-framework / lib / rex / proto / smb / client.rb

2103 lines (1631 loc) · 60.9 KB

Code Blame Raw

```ruby
# -*- coding: binary -*-
module Rex
module Proto
module SMB
class Client

  require 'net/ntlm'

require 'rex/text'
require 'rex/struct2'


# Some short-hand class aliases
CONST = Rex::Proto::SMB::Constants
CRYPT = Rex::Proto::SMB::Crypt
UTILS = Rex::Proto::SMB::Utils
XCEPT = Rex::Proto::SMB::Exceptions
EVADE = Rex::Proto::SMB::Evasions
NTLM_CRYPT = Rex::Proto::NTLM::Crypt
NTLM_CONST = Rex::Proto::NTLM::Constants
NTLM_UTILS = Rex::Proto::NTLM::Utils

  def initialize(socket)
    self.socket = socket
    self.native_os = 'Windows 2000 2195'
    self.native_lm = 'Windows 2000 5.0'
```

```
      self.encrypt_passwords = true
      self.extended_security = false
      self.multiplex_id = rand(0xffff)
      self.process_id = rand(0xffff)
      self.read_timeout = 10
      self.evasion_opts = {

        # Padding is performed between packet headers and data
        'pad_data' => EVADE::EVASION_NONE,

        # File path padding is performed on all open/create calls
        'pad_file' => EVADE::EVASION_NONE,

        # Modify the \PIPE\ string in trans_named_pipe calls
        'obscure_trans_pipe' => EVADE::EVASION_NONE,
      }

      self.verify_signature = false
      self.use_ntlmv2 = false
      self.usentlm2_session = true
      self.send_lm = true
      self.use_lanman_key = false
      self.send_ntlm  = true

      # Signing
      self.sequence_counter     = 0
      self.signing_key          = ''
      self.require_signing      = false
      self.peer_require_signing = false

      #Misc
      self.spnopt = {}
      self.default_max_buffer_size = 0xffdf
    end

    # Read a SMB packet from the socket
    def smb_recv

      data = socket.timed_read(4, self.read_timeout)
      if (data.nil? or data.length < 4)
        raise XCEPT::NoReply
      end

      recv_len = data[2,2].unpack('n')[0]
      if (recv_len == 0)
        return data
      end

      recv_len += 4

      while (data.length != recv_len)
        buff = ''

        begin
          buff << self.socket.timed_read(recv_len - data.length, self.read_timeout)
        rescue Timeout::Error
        rescue
          raise XCEPT::ReadPacket
        end

        if (buff.nil? or buff.length == 0)
          raise XCEPT::ReadPacket
        end

        data << buff
      end

      #signing
      if self.require_signing && self.signing_key != ''
        if self.verify_signature
          raise XCEPT::IncorrectSigningError if not CRYPT::is_signature_correct?(self.signing_key,sel
        end
        self.sequence_counter += 1
      end

      return data


    end

    # Send a SMB packet down the socket
    def smb_send(data, evasion_level=0)
      # evasion_level is ignored, since real evasion happens
      # in the actual socket layer

      size = 0
      wait = 0

      #signing
      if self.require_signing && self.signing_key != ''
        data = CRYPT::sign_smb_packet(self.signing_key, self.sequence_counter, data)
        self.sequence_counter += 1
```

```
            self.sequence_counter += 1
```

```
              end

              $stderr.puts [e, e.get_error(e.error_code), search_path]

              raise e
            end
          end

        end

        files.uniq
      end
```

```ruby

    # Creates a new directory on the mounted tree
    def create_directory(name)
      parm = [0].pack('V') + name + "\x00"
      resp = trans2(CONST::TRANS2_CREATE_DIRECTORY, parm, '')
      resp
    end


    # Send SMB echo request
    def echo(do_recv = true)

      pkt = CONST::SMB_ECHO_RES_PKT.make_struct
      self.smb_defaults(pkt['Payload']['SMB'])

      pkt['Payload']['SMB'].v['Command'] = CONST::SMB_COM_ECHO
      pkt['Payload']['SMB'].v['Flags1'] = 18
      if self.require_signing
        pkt['Payload']['SMB'].v['Flags2'] = 0x2807
      else
        pkt['Payload']['SMB'].v['Flags2'] =  0x2801
      end

      pkt['Payload']['SMB'].v['TreeID'] = self.last_tree_id || 0xffff
      pkt['Payload']['SMB'].v['WordCount'] = 1
      pkt['Payload'].v['EchoCount'] = 1

      ret = self.smb_send(pkt.to_s)
      return ret if not do_recv
      return self.smb_recv_parse(CONST::SMB_COM_ECHO)
    end


  # public read/write methods
    attr_accessor :native_os, :native_lm, :encrypt_passwords, :extended_security, :read_timeout, :eva
    attr_accessor :verify_signature, :use_ntlmv2, :usentlm2_session, :send_lm, :use_lanman_key, :send
    attr_accessor :system_time, :system_zone
    attr_accessor :spnopt
    attr_accessor :default_max_buffer_size

  # public read methods
    attr_reader           :dialect, :session_id, :challenge_key, :peer_native_lm, :peer_native_os
    attr_reader           :default_domain, :default_name, :auth_user, :auth_user_id
    attr_reader           :multiplex_id, :last_tree_id, :last_file_id, :process_id, :last_search_id
    attr_reader           :dns_host_name, :dns_domain_name
    attr_reader           :security_mode, :server_guid
    attr_reader           :sequence_counter,:signing_key, :require_signing, :peer_require_signing

  # private write methods
    attr_writer           :dialect, :session_id, :challenge_key, :peer_native_lm, :peer_native_os
    attr_writer           :default_domain, :default_name, :auth_user, :auth_user_id
    attr_writer           :dns_host_name, :dns_domain_name
    attr_writer           :multiplex_id, :last_tree_id, :last_file_id, :process_id, :last_search_id
    attr_writer           :security_mode, :server_guid
    attr_writer           :sequence_counter,:signing_key, :require_signing, :peer_require_signing

    attr_accessor :socket

  end
  end
  end
  end
```