Executable File · 473 lines (402 loc) · 19.1 KB


```
#!/usr/bin/env python
# Impacket - Collection of Python classes for working with network protocols.
#
# Copyright (C) 2022 Fortra. All rights reserved.
#
# This software is provided under a slightly modified version
# of the Apache Software License. See the accompanying LICENSE file
# for more information.
#
# Description:
#   A similar approach to smbexec but executing commands through WMI.
#   Main advantage here is it runs under the user (has to be Admin)
#   account, not SYSTEM, plus, it doesn't generate noisy messages
#   in the event log that smbexec.py does when creating a service.
#   Drawback is it needs DCOM, hence, I have to be able to access
#   DCOM ports at the target machine.
#
# Author:
#   beto (@agsolino)
#
# Reference for:
#   DCOM
#

from __future__ import division
from __future__ import print_function
```

```
27      import sys
28      import os
29      import cmd
30      import argparse
31      import time
32      import logging
33      import ntpath
34      from base64 import b64encode
35
36      from impacket.examples import logger
37      from impacket.examples.utils import parse_target
38      from impacket import version
39      from impacket.smbconnection import SMBConnection, SMB_DIALECT, SMB2_DIALECT_002, SMB2_DIALECT_21
40      from impacket.dcerpc.v5.dcomrt import DCOMConnection, COMVERSION
41      from impacket.dcerpc.v5.dcom import wmi
42      from impacket.dcerpc.v5.dtypes import NULL
43      from impacket.krb5.keytab import Keytab
44      from six import PY2
45
46      OUTPUT_FILENAME = '__' + str(time.time())
47      CODEC = sys.stdout.encoding
48
49
50      class WMIEXEC:
51          def __init__(self, command='', username='', password='', domain='', hashes=None, aesKey=None, s
52                       noOutput=False, doKerberos=False, kdcHost=None, shell_type=None):
53              self.__command = command
54              self.__username = username
55              self.__password = password
56              self.__domain = domain
57              self.__lmhash = ''
58              self.__nthash = ''
59              self.__aesKey = aesKey
60              self.__share = share
61              self.__noOutput = noOutput
62              self.__doKerberos = doKerberos
63              self.__kdcHost = kdcHost
64              self.__shell_type = shell_type
65              self.shell = None
66              if hashes is not None:
67                  self.__lmhash, self.__nthash = hashes.split(':')
68
69          def run(self, addr, silentCommand=False):
70              if self.__noOutput is False and silentCommand is False:
71                  smbConnection = SMBConnection(addr, addr)
72                  if self.__doKerberos is False:
```

```
                    smbConnection.login(self.__username, self.__password, self.__domain, self.__lmhash,
                else:
                    smbConnection.kerberosLogin(self.__username, self.__password, self.__domain, self._
                                                self.__nthash, self.__aesKey, kdcHost=self.__kdcHost)

                dialect = smbConnection.getDialect()
                if dialect == SMB_DIALECT:
                    logging.info("SMBv1 dialect used")
                elif dialect == SMB2_DIALECT_002:
                    logging.info("SMBv2.0 dialect used")
                elif dialect == SMB2_DIALECT_21:
                    logging.info("SMBv2.1 dialect used")
                else:
                    logging.info("SMBv3.0 dialect used")
            else:
                smbConnection = None

            dcom = DCOMConnection(addr, self.__username, self.__password, self.__domain, self.__lmhash,
                                  self.__aesKey, oxidResolver=True, doKerberos=self.__doKerberos, kdcHo
            try:
                iInterface = dcom.CoCreateInstanceEx(wmi.CLSID_WbemLevel1Login, wmi.IID_IWbemLevel1Logi
                iWbemLevel1Login = wmi.IWbemLevel1Login(iInterface)
                iWbemServices = iWbemLevel1Login.NTLMLogin('//./root/cimv2', NULL, NULL)
                iWbemLevel1Login.RemRelease()

                win32Process, _ = iWbemServices.GetObject('Win32_Process')

                self.shell = RemoteShell(self.__share, win32Process, smbConnection, self.__shell_type,
                if self.__command != ' ':
                    self.shell.onecmd(self.__command)
                else:
                    self.shell.cmdloop()
            except  (Exception, KeyboardInterrupt) as e:
                if logging.getLogger().level == logging.DEBUG:
                    import traceback
                    traceback.print_exc()
                logging.error(str(e))
                if smbConnection is not None:
                    smbConnection.logoff()
                dcom.disconnect()
                sys.stdout.flush()
                sys.exit(1)

            if smbConnection is not None:
                smbConnection.logoff()
            dcom.disconnect()
```

```
118                 dcom.disconnect()
```

```
400         if len(sys.argv) == 1:
401             parser.print_help()
402             sys.exit(1)
403 
404         options = parser.parse_args()
405 
406         # Init the example's logger theme
407         logger.init(options.ts)
408 
409         if options.codec is not None:
410             CODEC = options.codec
411         else:
412             if CODEC is None:
413                 CODEC = 'utf-8'
414 
415         if ' '.join(options.command) == ' ' and options.nooutput is True:
416             logging.error("-nooutput switch and interactive shell not supported")
417             sys.exit(1)
418         if options.silentcommand and options.command == ' ':
419             logging.error("-silentcommand switch and interactive shell not supported")
420             sys.exit(1)
421 
422         if options.debug is True:
423             logging.getLogger().setLevel(logging.DEBUG)
424             # Print the Library's installation path
425             logging.debug(version.getInstallationPath())
426         else:
427             logging.getLogger().setLevel(logging.INFO)
428 
429         if options.com_version is not None:
430             try:
431                 major_version, minor_version = options.com_version.split('.')
432                 COMVERSION.set_default_version(int(major_version), int(minor_version))
433             except Exception:
434                 logging.error("Wrong COMVERSION format, use dot separated integers e.g. \"5.7\"")
435                 sys.exit(1)
436 
437         domain, username, password, address = parse_target(options.target)
438 
```

```
439         try:
440             if options.A is not None:
441                 (domain, username, password) = load_smbclient_auth_file(options.A)
442                 logging.debug('loaded smbclient auth file: domain=%s, username=%s, password=%s' % (
443                 repr(domain), repr(username), repr(password)))
444 
445             if domain is None:
446                 domain = ''
447 
448             if options.keytab is not None:
449                 Keytab.loadKeysFromKeytab(options.keytab, username, domain, options)
450                 options.k = True
451 
452             if password == '' and username != '' and options.hashes is None and options.no_pass is Fals
453                 from getpass import getpass
454 
455                 password = getpass("Password:")
456 
457             if options.aesKey is not None:
458                 options.k = True
459 
460             executer = WMIEXEC(' '.join(options.command), username, password, domain, options.hashes, c
461                                options.share, options.nooutput, options.k, options.dc_ip, options.shell
462             executer.run(address, options.silentcommand)
463         except KeyboardInterrupt as e:
464             logging.error(str(e))
465         except Exception as e:
466             if logging.getLogger().level == logging.DEBUG:
467                 import traceback
468 
469                 traceback.print_exc()
470             logging.error(str(e))
471             sys.exit(1)
472 
473         sys.exit(0)
```