impacket / examples / atexec.py

Executable File · 319 lines (276 loc) · 12.6 KB

Code Blame Raw

```
#!/usr/bin/env python
# Impacket - Collection of Python classes for working with network protocols.
#
# SECUREAUTH LABS. Copyright (C) 2021 SecureAuth Corporation. 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:
#   ATSVC example for some functions implemented, creates, enums, runs, delete jobs
#   This example executes a command on the target machine through the Task Scheduler
#   service. Returns the output of such command
#
# Author:
#   Alberto Solino (@agsolino)
#
# Reference for:
#   DCE/RPC for TSCH
#

from __future__ import division
from __future__ import print_function
import string
import sys
import argparse
```

```
27      import time
28      import random
29      import logging
30
31      from impacket.examples import logger
32      from impacket import version
33      from impacket.dcerpc.v5 import tsch, transport
34      from impacket.dcerpc.v5.dtypes import NULL
35      from impacket.dcerpc.v5.rpcrt import RPC_C_AUTHN_GSS_NEGOTIATE, \
36          RPC_C_AUTHN_LEVEL_PKT_PRIVACY
37      from impacket.examples.utils import parse_target
38      from impacket.krb5.keytab import Keytab
39      from six import PY2
40
41      CODEC = sys.stdout.encoding
42
43      class TSCH_EXEC:
44          def __init__(self, username='', password='', domain='', hashes=None, aesKey=None, doKerberos=Fa
45                       command=None, sessionId=None, silentCommand=False):
46              self.__username = username
47              self.__password = password
48              self.__domain = domain
49              self.__lmhash = ''
50              self.__nthash = ''
51              self.__aesKey = aesKey
52              self.__doKerberos = doKerberos
53              self.__kdcHost = kdcHost
54              self.__command = command
55              self.__silentCommand = silentCommand
56              self.sessionId = sessionId
57
58              if hashes is not None:
59                  self.__lmhash, self.__nthash = hashes.split(':')
60
61          def play(self, addr):
62              stringbinding = r'ncacn_np:%s[\pipe\atsvc]' % addr
63              rpctransport = transport.DCERPCTransportFactory(stringbinding)
64
65              if hasattr(rpctransport, 'set_credentials'):
66                  # This method exists only for selected protocol sequences.
67                  rpctransport.set_credentials(self.__username, self.__password, self.__domain, self.__lm
68                                               self.__aesKey)
69                  rpctransport.set_kerberos(self.__doKerberos, self.__kdcHost)
70              try:
71                  self.doStuff(rpctransport)
72              except Exception as e:
```

```
73                  if logging.getLogger().level == logging.DEBUG:
74                      import traceback
75                      traceback.print_exc()
76                  logging.error(e)
77                  if str(e).find('STATUS_OBJECT_NAME_NOT_FOUND') >=0:
78                      logging.info('When STATUS_OBJECT_NAME_NOT_FOUND is received, try running again. It
79
80      def doStuff(self, rpctransport):
81          def output_callback(data):
82              try:
83                  print(data.decode(CODEC))
84              except UnicodeDecodeError:
85                  logging.error('Decoding error detected, consider running chcp.com at the target,\nm
86                                'https://docs.python.org/3/library/codecs.html#standard-encodings\nar
87                                'again with -codec and the corresponding codec')
88                  print(data.decode(CODEC, errors='replace'))
89
90          def xml_escape(data):
91              replace_table = {
92                   "&": "&amp;",
93                   '"': "&quot;",
94                   "'": "&apos;",
95                   ">": "&gt;",
96                   "<": "&lt;",
97                   }
98              return ''.join(replace_table.get(c, c) for c in data)
99
100         def cmd_split(cmdline):
101             cmdline = cmdline.split(" ", 1)
102             cmd = cmdline[0]
103             args = cmdline[1] if len(cmdline) > 1 else ''
104
105             return [cmd, args]
106
107         dce = rpctransport.get_dce_rpc()
108
109         dce.set_credentials(*rpctransport.get_credentials())
110         if self.__doKerberos is True:
111             dce.set_auth_type(RPC_C_AUTHN_GSS_NEGOTIATE)
112         dce.connect()
113         dce.set_auth_level(RPC_C_AUTHN_LEVEL_PKT_PRIVACY)
114         dce.bind(tsch.MSRPC_UUID_TSCHS)
115         tmpName = ''.join([random.choice(string.ascii_letters) for _ in range(8)])
116         tmpFileName = tmpName + '.tmp'
117
118         if self.sessionId is not None:
```

```
246         parser.add_argument('target', action='store', help='[[domain/]username[:password]@]<targetName
247         parser.add_argument('command', action='store', nargs='*', default=' ', help='command to execute
248         parser.add_argument('-session-id', action='store', type=int, help='an existed logon session to
249         parser.add_argument('-ts', action='store_true', help='adds timestamp to every logging output')
250         parser.add_argument('-silentcommand', action='store_true', default = False, help='does not exec
251                                                                                 'given command
252         parser.add_argument('-debug', action='store_true', help='Turn DEBUG output ON')
253         parser.add_argument('-codec', action='store', help='Sets encoding used (codec) from the target\
254                                                            '"%s"). If errors are detected, run chcp.com
255                                                            'map the result with '
```

```
256                             'https://docs.python.org/3/library/codecs.html#standard-encodings and the
257                             'again with -codec and the corresponding codec ' % CODEC)
258
259         group = parser.add_argument_group('authentication')
260
261         group.add_argument('-hashes', action="store", metavar = "LMHASH:NTHASH", help='NTLM hashes, for
262         group.add_argument('-no-pass', action="store_true", help='don\'t ask for password (useful for -
263         group.add_argument('-k', action="store_true", help='Use Kerberos authentication. Grabs credenti
264                            '(KRB5CCNAME) based on target parameters. If valid credentials cannot be fou
265                            'ones specified in the command line')
266         group.add_argument('-aesKey', action="store", metavar = "hex key", help='AES key to use for Ker
267                                                                                 '(128 or 256 bits)')
268         group.add_argument('-dc-ip', action='store',metavar = "ip address",  help='IP Address of the dc
269                                              'If omitted it will use the domain part (FQDN) specified i
270         group.add_argument('-keytab', action="store", help='Read keys for SPN from keytab file')
271
272         if len(sys.argv)==1:
273             parser.print_help()
274             sys.exit(1)
275
276         options = parser.parse_args()
277
278         # Init the example's logger theme
279         logger.init(options.ts)
280
281         if options.codec is not None:
282             CODEC = options.codec
283         else:
284             if CODEC is None:
285                 CODEC = 'utf-8'
286
287         logging.warning("This will work ONLY on Windows >= Vista")
288
289         if ''.join(options.command) == ' ':
290             logging.error('You need to specify a command to execute!')
291             sys.exit(1)
292
293         if options.debug is True:
294             logging.getLogger().setLevel(logging.DEBUG)
295             # Print the Library's installation path
296             logging.debug(version.getInstallationPath())
297         else:
298             logging.getLogger().setLevel(logging.INFO)
299
300         domain, username, password, address = parse_target(options.target)
301
```

```
302         if domain is None:
303             domain = ''
304 
305         if options.keytab is not None:
306             Keytab.loadKeysFromKeytab (options.keytab, username, domain, options)
307             options.k = True
308 
309         if password == '' and username != '' and options.hashes is None and options.no_pass is False ar
310             from getpass import getpass
311 
312             password = getpass("Password:")
313 
314         if options.aesKey is not None:
315             options.k = True
316 
317         atsvc_exec = TSCH_EXEC(username, password, domain, options.hashes, options.aesKey, options.k, c
318                                ' '.join(options.command), options.session_id, options.silentcommand)
319         atsvc_exec.play(address)
```