Files

8b1a99f

Go to file

- .github
- examples
  - Get-GPPPassword.py
  - GetADUsers.py
  - GetNPUsers.py
  - GetUserSPNs.py
  - addcomputer.py
  - atexec.py
  - dcomexec.py
  - dpapi.py
  - esentutl.py
  - exchanger.py
  - findDelegation.py
  - getArch.py
  - getPac.py
  - getST.py
  - getTGT.py
  - goldenPac.py
  - karmaSMB.py
  - keylistattack.py
  - kintercept.py
  - lookupsid.py
  - machine_role.py
  - mimikatz.py
  - mqtt_check.py
  - mssqlclient.py
  - mssqlinstance.py
  - netview.py
  - nmapAnswerMachine.py
  - ntfs-read.py
  - ntlmrelayx.py
  - ping.py
  - ping6.py
  - psexec.py
  - raiseChild.py
  - rbcd.py

impacket / examples / atexec.py

martingalloar Arrange tagline, copyright and license notes across... cd4fe47 · 3 years ago History

Code Blame Executable File · 319 lines (276 loc) · 12.6 KB Raw

```python
#!/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
import time
import random
import logging

from impacket.examples import logger
from impacket import version
from impacket.dcerpc.v5 import tsch, transport
from impacket.dcerpc.v5.dtypes import NULL
from impacket.dcerpc.v5.rpcrt import RPC_C_AUTHN_GSS_NEGOTIATE, \
    RPC_C_AUTHN_LEVEL_PKT_PRIVACY
from impacket.examples.utils import parse_target
from impacket.krb5.keytab import Keytab
from six import PY2

CODEC = sys.stdout.encoding

class TSCH_EXEC:
    def __init__(self, username='', password='', domain='', hashes=None, aesKey=None, d
                 command=None, sessionId=None, silentCommand=False):
        self.__username = username
        self.__password = password
        self.__domain = domain
        self.__lmhash = ''
        self.__nthash = ''
        self.__aesKey = aesKey
        self.__doKerberos = doKerberos
        self.__kdcHost = kdcHost
        self.__command = command
        self.__silentCommand = silentCommand
        self.sessionId = sessionId
```

- rdp_check.py
- reg.py
- registry-read.py
- rpcdump.py
- rpcmap.py
- sambaPipe.py

```
        if hashes is not None:
            self.__lmhash, self.__nthash = hashes.split(':')

    def play(self, addr):
        stringbinding = r'ncacn_np:%s[\pipe\atsvc]' % addr
        rpctransport = transport.DCERPCTransportFactory(stringbinding)

        if hasattr(rpctransport, 'set_credentials'):
            # This method exists only for selected protocol sequences.
            rpctransport.set_credentials(self.__username, self.__password, self.__domai
                                         self.__aesKey)
            rpctransport.set_kerberos(self.__doKerberos, self.__kdcHost)
        try:
            self.doStuff(rpctransport)
        except Exception as e:
            if logging.getLogger().level == logging.DEBUG:
                import traceback
                traceback.print_exc()
            logging.error(e)
            if str(e).find('STATUS_OBJECT_NAME_NOT_FOUND') >=0:
                logging.info('When STATUS_OBJECT_NAME_NOT_FOUND is received, try runnin

    def doStuff(self, rpctransport):
        def output_callback(data):
            try:
                print(data.decode(CODEC))
            except UnicodeDecodeError:
                logging.error('Decoding error detected, consider running chcp.com at th
                              'https://docs.python.org/3/library/codecs.html#standard-e
                              'again with -codec and the corresponding codec')
                print(data.decode(CODEC, errors='replace'))

        def xml_escape(data):
            replace_table = {
                 "&": "&amp;",
                 '"': "&quot;",
                 "'": "&apos;",
                 ">": "&gt;",
                 "<": "&lt;",
                 }
            return ''.join(replace_table.get(c, c) for c in data)

        def cmd_split(cmdline):
            cmdline = cmdline.split(" ", 1)
            cmd = cmdline[0]
            args = cmdline[1] if len(cmdline) > 1 else ''

            return [cmd, args]

        dce = rpctransport.get_dce_rpc()

        dce.set_credentials(*rpctransport.get_credentials())
        if self.__doKerberos is True:
            dce.set_auth_type(RPC_C_AUTHN_GSS_NEGOTIATE)
        dce.connect()
        dce.set_auth_level(RPC_C_AUTHN_LEVEL_PKT_PRIVACY)
        dce.bind(tsch.MSRPC_UUID_TSCHS)
        tmpName = ''.join([random.choice(string.ascii_letters) for _ in range(8)])
        tmpFileName = tmpName + '.tmp'

        if self.sessionId is not None:
```

```
    parser.add_argument('target', action='store', help='[[domain/]username[:password]@]
    parser.add_argument('command', action='store', nargs='*', default=' ', help='comman
    parser.add_argument('-session-id', action='store', type=int, help='an existed logon
    parser.add_argument('-ts', action='store_true', help='adds timestamp to every loggi
    parser.add_argument('-silentcommand', action='store_true', default = False, help='d
                                                                                  'g
    parser.add_argument('-debug', action='store_true', help='Turn DEBUG output ON')
    parser.add_argument('-codec', action='store', help='Sets encoding used (codec) from
                                                       '"%s"). If errors are detected,
                                                       'map the result with '
                          'https://docs.python.org/3/library/codecs.html#standard-encod
                          'again with -codec and the corresponding codec ' % CODEC)

    group = parser.add_argument_group('authentication')

    group.add_argument('-hashes', action="store", metavar = "LMHASH:NTHASH", help='NTLM
    group.add_argument('-no-pass', action="store_true", help='don\'t ask for password (
    group.add_argument('-k', action="store_true", help='Use Kerberos authentication. Gr
                       '(KRB5CCNAME) based on target parameters. If valid credentials c
                       'ones specified in the command line')
    group.add_argument('-aesKey', action="store", metavar = "hex key", help='AES key to
                                                                               '(128 or 25
    group.add_argument('-dc-ip', action='store',metavar = "ip address",  help='IP Addre
                                                 'If omitted it will use the domain part (FQDN)
    group.add_argument('-keytab', action="store", help='Read keys for SPN from keytab f

    if len(sys.argv)==1:
        parser.print_help()
        sys.exit(1)

    options = parser.parse_args()

    # Init the example's logger theme
    logger.init(options.ts)

    if options.codec is not None:
        CODEC = options.codec
    else:
        if CODEC is None:
            CODEC = 'utf-8'

    logging.warning("This will work ONLY on Windows >= Vista")

    if ''.join(options.command) == ' ':
        logging.error('You need to specify a command to execute!')
        sys.exit(1)

    if options.debug is True:
        logging.getLogger().setLevel(logging.DEBUG)
        # Print the Library's installation path
        logging.debug(version.getInstallationPath())
    else:
        logging.getLogger().setLevel(logging.INFO)

    domain, username, password, address = parse_target(options.target)

    if domain is None:
        domain = ''

    if options.keytab is not None:
        Keytab.loadKeysFromKeytab (options.keytab, username, domain, options)
        options.k = True

    if password == '' and username != '' and options.hashes is None and options.no_pass

        from getpass import getpass

        password = getpass("Password:")

    if options.aesKey is not None:
        options.k = True

    atsvc_exec = TSCH_EXEC(username, password, domain, options.hashes, options.aesKey,
                           ' '.join(options.command), options.session_id, options.silen
    atsvc_exec.play(address)
```