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 / smbexec.py

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

Code Blame Executable File · 404 lines (347 loc) · 16.2 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:
#   A similar approach to psexec w/o using RemComSvc. The technique is described here
#   https://www.optiv.com/blog/owning-computers-without-shell-access
#   Our implementation goes one step further, instantiating a local smbserver to receiv
#   output of the commands. This is useful in the situation where the target machine do
#   have a writeable share available.
#   Keep in mind that, although this technique might help avoiding AVs, there are a lot
#   event logs generated and you can't expect executing tasks that will last long since
#   will kill the process since it's not responding as a Windows service.
#   Certainly not a stealthy way.
#
#   This script works in two ways:
#       1) share mode: you specify a share, and everything is done through that share.
#       2) server mode: if for any reason there's no share available, this script will
#          SMB server, so the output of the commands executed are sent back by the targ
#          into a locally shared folder. Keep in mind you would need root access to bin
#          in the local machine.
#
# Author:
#   beto (@agsolino)
#
# Reference for:
#   DCE/RPC and SMB.
#

from __future__ import division
from __future__ import print_function
import sys
import os
import cmd
import argparse
try:
    import ConfigParser
except ImportError:
    import configparser as ConfigParser
import logging
from threading import Thread
from base64 import b64encode

from impacket.examples import logger
from impacket.examples.utils import parse_target
from impacket import version, smbserver
from impacket.dcerpc.v5 import transport, scmr
from impacket.krb5.keytab import Keytab

OUTPUT_FILENAME = '__output'
BATCH_FILENAME  = 'execute.bat'
```

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

```python
SMBSERVER_DIR   = '__tmp'
DUMMY_SHARE     = 'TMP'
SERVICE_NAME    = 'BTOBTO'
CODEC = sys.stdout.encoding

class SMBServer(Thread):
    def __init__(self):
        Thread.__init__(self)
        self.smb = None

    def cleanup_server(self):
        logging.info('Cleaning up..')
        try:
            os.unlink(SMBSERVER_DIR + '/smb.log')
        except OSError:
            pass
        os.rmdir(SMBSERVER_DIR)

    def run(self):
        # Here we write a mini config for the server
        smbConfig = ConfigParser.ConfigParser()
        smbConfig.add_section('global')
        smbConfig.set('global','server_name','server_name')
        smbConfig.set('global','server_os','UNIX')
        smbConfig.set('global','server_domain','WORKGROUP')
        smbConfig.set('global','log_file',SMBSERVER_DIR + '/smb.log')
        smbConfig.set('global','credentials_file','')

        # Let's add a dummy share
        smbConfig.add_section(DUMMY_SHARE)
        smbConfig.set(DUMMY_SHARE,'comment','')
        smbConfig.set(DUMMY_SHARE,'read only','no')
        smbConfig.set(DUMMY_SHARE,'share type','0')
        smbConfig.set(DUMMY_SHARE,'path',SMBSERVER_DIR)

        # IPC always needed
        smbConfig.add_section('IPC$')
        smbConfig.set('IPC$','comment','')
        smbConfig.set('IPC$','read only','yes')
        smbConfig.set('IPC$','share type','3')
        smbConfig.set('IPC$','path')

        self.smb = smbserver.SMBSERVER(('0.0.0.0',445), config_parser = smbConfig)
        logging.info('Creating tmp directory')
        try:
            os.mkdir(SMBSERVER_DIR)
        except Exception as e:
            logging.critical(str(e))
            pass
        logging.info('Setting up SMB Server')
        self.smb.processConfigFile()
        logging.info('Ready to listen...')
        try:
            self.smb.serve_forever()
        except:
            pass

    def stop(self):
        self.cleanup_server()
        self.smb.socket.close()
        self.smb.server_close()
        self._Thread__stop()
```

```

    group.add_argument('-dc-ip', action='store',metavar = "ip address", help='IP Addres
                       'If omitted it will use the domain part (FQDN) specified in the
    group.add_argument('-target-ip', action='store', metavar="ip address", help='IP Add
                       'ommited it will use whatever was specified as target. This is u
                       'name and you cannot resolve it')
    group.add_argument('-port', choices=['139', '445'], nargs='?', default='445', metav
                       help='Destination port to connect to SMB Server')
    group.add_argument('-service-name', action='store', metavar="service_name", default
                                        'service used to trigger the payload')

    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('-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'

        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, remoteName = 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.target_ip is None:
            options.target_ip = remoteName

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

        try:
            executer = CMDEXEC(username, password, domain, options.hashes, options.aesKey,
                               options.mode, options.share, int(options.port), options.serv
            executer.run(remoteName, options.target_ip)
        except Exception as e:
            if logging.getLogger().level == logging.DEBUG:
                import traceback
                traceback.print_exc()
            logging.critical(str(e))
        sys.exit(0)
```